

AD-A061 636

SYRACUSE UNIV N Y
MULTILEVEL MODULARIZATION OF SYSTEMS TO MINIMIZE LIFE CYCLE COS--ETC(U)
SEP 78 J E BIEGEL, B BULCHA

F/G 9/3
F30602-75-C-0121

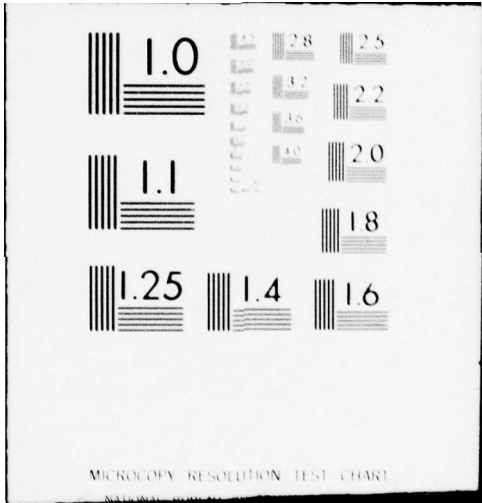
UNCLASSIFIED

RADC-TR-78-207

NL

1 of 1
AD
A061 636





MICROCOPY RESOLUTION TEST CHART

LEVEL

II

12

AD A061 636

RADC-TR-78-207
Final Technical Report
September 1978



MULTILEVEL MODULARIZATION OF SYSTEMS TO MINIMIZE LIFE CYCLE COST

John E. Biegel
Bisrat Bulcha

Syracuse University

DDC FILE COPY

Approved for public release; distribution unlimited.

ROME AIR DEVELOPMENT CENTER
Air Force Systems Command
Griffiss Air Force Base, New York 13441

DDC
RECEIVED
NOV 29 1978
REGULATED

Handwritten mark

D

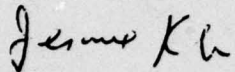
78 11 20 002

This report contains a large percentage of machine-produced copy which is not of the highest printing quality but because of economical consideration, it was determined in the best interest of the government that they be used in this publication.

This report has been reviewed by the RADC Information Office (OI) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

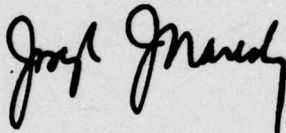
RADC-TR-78-207 has been reviewed and is approved for publication.

APPROVED:



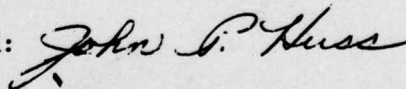
JEROME KLION
Project Engineer

APPROVED:



JOSEPH J. NARESKY
Chief, Reliability & Compatibility Division

FOR THE COMMANDER:



JOHN P. HUSS
Acting Chief, Plans Office

If your address has changed or if you wish to be removed from the RADC mailing list, or if the addressee is no longer employed by your organization, please notify RADC (RBRT) Griffiss AFB NY 13441. This will assist us in maintaining a current mailing list.

Do not return this copy. Retain or destroy.

MISSION
of
Rome Air Development Center

RADC plans and conducts research, exploratory and advanced development programs in command, control, and communications (C³) activities, and in the C³ areas of information sciences and intelligence. The principal technical mission areas are communications, electromagnetic guidance and control, surveillance of ground and aerospace objects, intelligence data collection and handling, information system technology, ionospheric propagation, solid state sciences, microwave physics and electronic reliability, maintainability and compatibility.



UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

19 REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM	
18 1. REPORT NUMBER RADCR-78-207	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER	9
6 4. TITLE (and Subtitle) MULTILEVEL MODULARIZATION OF SYSTEMS TO MINIMIZE LIFE CYCLE COST	5. TYPE OF REPORT & PERIOD COVERED Final Technical Report		
10 7. AUTHOR(s) John E. Biegel Bisrat Bulcha	15 8. CONTRACT OR GRANT NUMBER(s) F30602-75-C-0121	6. PERFORMING ORG. REPORT NUMBER N/A	
9. PERFORMING ORGANIZATION NAME AND ADDRESS Syracuse University Syracuse NY 13210	16 10. PROGRAM ELEMENT PROJECT, TASK AREA & WORK UNIT NUMBERS 62702F 95670016	11 13. REPORT DATE September 78	14 13. NUMBER OF PAGES 57
11. CONTROLLING OFFICE NAME AND ADDRESS Rome Air Development Center (RBRT) Griffiss AFB NY 13441	12 14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) Same	15. SECURITY CLASS. (of this report) UNCLASSIFIED	15a. DECLASSIFICATION DOWNGRADING SCHEDULE N/A
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) Same			
18. SUPPLEMENTARY NOTES RADCR Project Engineer: Jerome Klion (RBRT)			
19. KEY WORDS (Continue on reverse side if necessary and identify by block number) Availability Life Cycle Cost Reliability Math Model			
20. ABSTRACT (Continue on reverse side if necessary and identify by block number) Every electronic equipment modularization (partitioning of an electronic equipment into different numbers of line replaceable units) scheme incurs a different life cycle cost. Methodology which allows an equipment to be modularized or partitioned in a fashion such that life cycle cost is minimized must be developed. This report documents a new procedure which can be used to this end.			

DD FORM 1 JAN 73 1473

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

339 600

78

11

20

002

Lu

Preface

This report describes a technique for the multi-level modularization of large systems such that the life cycle cost will be a minimum. The method presented is an extension of the one-level modularization reported in Biegel and Bulcha, System Modularization to Minimize Life Cycle Costs [1,2].

The method is a heuristic extension of the previous method [1,2] and no attempt has been made to prove its optimality. We have generated no counter examples, however.

The system is first decomposed into functional elements, then reconstructed into modules, each containing one or more functional elements. The modules are then collected into subassemblies, the subassemblies into higher level subassemblies, etc. The criteria is to form the collected sets in such a way that the life cycle cost (LCC) is minimized.

The computer routine to do this is presented and explained. An example problem is included.

ACCESSION BY		
DTIC	White Sands <input checked="" type="checkbox"/>	
DDC	DDP Seattle <input type="checkbox"/>	
UNANNOUNCED	<input type="checkbox"/>	
JUSTIFICATION		
BY		
DISTRIBUTION/AVAILABILITY CODES		
SICL	AVAIL. CODE/WT	FORMAL
A		

LEVEL II

DDC
RECEIVED
NOV 29 1978
D

EVALUATION

The objective of this effort was to develop a methodology which would allow an equipment to be modularized (partitioned into modules) in such a way as to minimize total life cycle cost. This objective was accomplished. The methodology developed is capable of structuring the modular organization of an equipment taking into account reliability, maintainability, fabrication costs, and logistics support costs. This report provides the procedures and necessary detail for use of this methodology. Besides the dissemination of the report to potential users, follow-on activity is currently in progress to:

- a. Include the methodology in a planned "Life Cycle Cost Design Handbook"
- b. Include the methodology as part of the "RADC Compu-Standards Program". (A computerized compendium of procedures intended to implement and support reliability and maintainability standards and handbooks).
- c. Utilize the methodology in the house and suggest its use to RADC contractors in support of life cycle cost analysis efforts on hardware items.


JEROME KLION
Project Engineer

I. INTRODUCTION

The objective of this research was to develop a technique for the multi-level modularization of large systems through operations on their matrix representation. The final procedure must perform m levels of modularization on a large network of n nodes such that the life cycle cost will be at or near minimum. The solution technique is programmed into the RADC Multics system in Fortran. This is an extension of the work reported in Biegel and Bulcha, System Modularization to Minimize Life Cycle Costs [1,2].

As electrical designs become increasingly complex and large, so does their network representation, and the associated data describing the network and its components. Hence in developing an efficient partitioning technique for large networks (100 or more nodes) data handling becomes a major consideration.

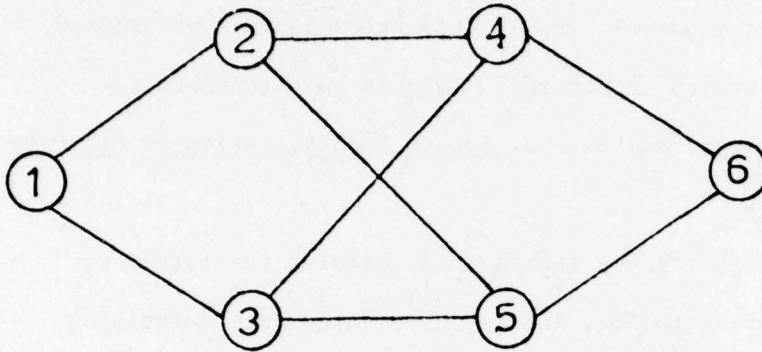
Network information is conveniently represented in matrix form. But as the number of nodes in the network increases, so does the matrix size. If there are 100 nodes, one needs a (100 X 100) matrix for a full representation of the network.

Although the modularization procedure does not assume any particular type of network, it is necessary to look more closely into different types of networks from the data handling point of view.

The modularization process described in this report is essentially the same as that described in our previous report [1,2]. We have incorporated a computerized spares allocation procedure and the complete life cycle cost evaluation of the designs developed when modularizing these large networks. In many engineering applications, including electronics, the network that represents the system is of an elongated type and the matrix is called "sparse".

Definition: Let A be a square matrix of order n , if r is the number of non-zero elements and $r \ll n^2$, then A is sparse.

An Elongated Non-Directed Network



Its Matrix Representation

	1	2	3	4	5	6
1	0	1	1	0	0	0
2	1	0	0	1	1	0
3	1	0	0	1	1	0
4	0	1	1	0	0	1
5	0	1	1	0	0	1
6	0	0	0	1	1	0

Figure 1: An elongated network and it's sparse matrix representation.

Now consider a non-directed network that results in a matrix that is dense, typically a "complete graph".

Definition: A complete graph, is a graph where every node is connected to every other node.

	1	2	3	4
1	0	1	1	1
2	1	0	1	1
3	1	1	0	1
4	1	1	1	0

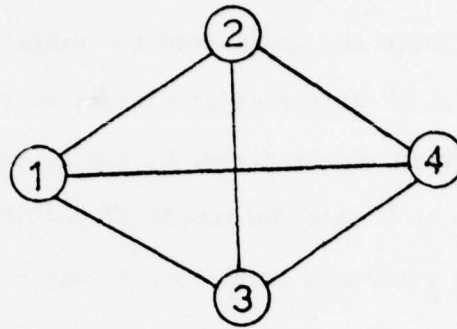


Figure 2: A complete graph and its dense matrix representation.

As the number of nodes increases, the proportion of non-zero entries to the size of the matrix is usually significantly reduced. Thus, efficient methods of storing data for large networks must be found.

Elongated, directed networks have a triangular matrix representation where data is usually concentrated near the diagonal. For such a matrix, where $n = 100$, we have up to 4950 possible entries and only a small percentage of these entries are non-zero. The usual way of specifying an entry by the row and column indices is inefficient and time and space consuming when working with large networks. Considering this difficulty, a procedure is developed to specify for each row only those elements to the immediate right of the main diagonal up to and including the last non-zero element in that row. For example, consider the i^{th} row with elements $A_i, A_{i+1}, A_{i+2}, \dots, A_i, n$. Under the procedure used in this program, those elements would be entered as: $j, k, 0, 0, 0, 1, 0, m$ where j, k, l and m are the only non-zero elements in the i^{th} row with j being the element in the $(i + 1)$ st column. This information is input as above and used in a subroutine, INIT, where the complete $n \times n$ matrix is created for use in the modularization procedure in the main program.

II. ON THE MODULARIZATION OF LARGE NETWORKS: GENERATION OF PROPER CUTS

An exhaustive enumeration approach can be used to generate all possible sets S and \bar{S} . We can test if these sets satisfy the requirement of a proper cut (divide the graph into two subgraphs). Clearly for a graph of n nodes there are 2^n different partitions of the nodes into the set S and \bar{S} . Moreover if it is required that U , the source node, be in S and V , the sink node, be in \bar{S} then the number of partitions reduces to 2^{n-2} . Obviously, the number of cuts in a graph is not only a function of the number of nodes n but depends on the configuration of the network as well. For a simple chain type network the number of cuts is $n-1$. For a completely connected graph, the upper bound is the limit given above which is 2^{n-2} . For large n , this results in a very wide range in the number of cuts.

Two heuristic methods directed toward reducing the number of cuts to be generated are:

1. to generate only those cuts that are connected, and
2. to generate restricted proper cuts.

A connected cut is one in which all nodes in a set are connected by a chain. A restricted proper cut is one such that the source node is in S and the sink node is in \bar{S} .

If a network has as many as 100 nodes, it is highly doubtful that these or any other similar cut generation procedures will adequately reduce the number of sets to be considered in an optimization process. The procedure developed during this research limits the number of cuts to those which have a high probability of providing an optimal result.

Paul A. Jensen *states "...any algorithm for generating cuts will soon be time limited in operation as one increases the size and complexity of the subject graphs. On this basis one should be suspect of the practicality for all but the smallest problem of an optimization procedure which requires for its performance the set of all proper cuts."

* Jensen, Paul A, A Graph Decomposition Technique for the Design of Reliable Redundant Electronic Networks, Ph.D. Dissertation, Johns Hopkins University, page 126.

III. THE MODULARIZATION PROCEDURE AT m-LEVELS:

After the basic modularization of a large network, the resulting configuration is again a network with the nodes replaced by a higher level nodes called modules and the arcs being the interconnections between these modules. Further "modularization" will result in still a higher form of modules that are frequently called subassemblies.

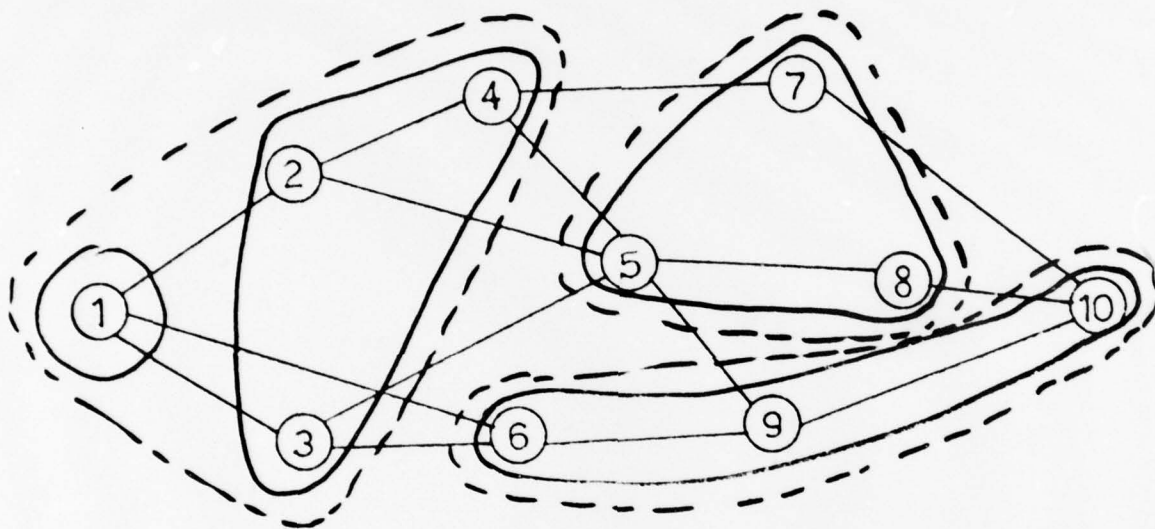


Figure 3: A network of subassemblies.

The procedure can be applied as many times as required to obtain the desired number of levels of modularization. The only modification in the procedure is in the network input data itself.

Consider a network that has been built up of modules. We are interested in generating higher level assemblies. The network information is changed straight forwardly as follows:

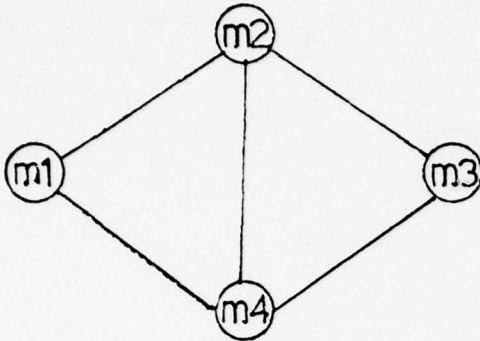


Figure 4: A network of modules.

Where $m_1 = [1]$, $m_2 = [2,3,4]$, $m_3 = [5,7,8]$, $m_4 = [6,9,10]$.

The intraconnection $I(M_i, M_j)$ between the newly found nodes, M_i and M_j can be obtained by forming all the pairs (k, m) such that $k \in M_i$ and $m \in M_j$, and entering the current network interconnection matrix, A_c to read $A_c(k, m)$. Then

$$I(M_i, M_j) = \sum_{\forall (k, m) \text{ pairs}} A_c(k, m).$$

Thus a new network is developed for yet another level of partitioning, if needed. If an m level modularization is wanted, this procedure will yield m new networks. At any level the newly found network and its characteristics are a new set of data for the modularization algorithm explained in Biegel and Bulcha [1,2].

It is conceivable that at some stage a user might want some of the nodes left at lower levels with the rest of the nodes merged into a higher level of assemblies. (This could be for maintenance reasons.) This variation can be incorporated by adding fictitious nodes to the network in place of the nodes that are not candidates for higher level assemblies. These fictitious nodes will have 0 arc weights and 0 physical characteristic in

the next level of network. Figure 5 shows such a network.

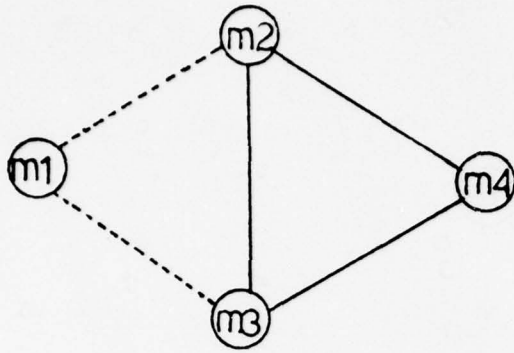


Figure 5: A network with a fictitious node.

Where M_k - fictitious for next level network i.e.:

$$\phi_{M_k}^j = 0 \quad \forall j; \quad I(M_1, M_2) = I(M_2, M_1) = 0, \quad I(M_1, M_3) = I(M_3, M_1) = 0$$

III. THE COST MODEL FOR HIGHER LEVEL DESIGNS INVOLVING SUBASSEMBLIES AND ASSEMBLIES

Suppose a design D consists of a mixture of modules (M), subassemblies (SA), and assemblies (SS). Assume that there are t modules at a modular level, m subassemblies and p assemblies at the final level as shown in the Figure 6.

The acquisition cost of such a design is:

$$C_A(D) = \sum_{i=1}^t C(M)_i + \sum_{j=1}^m C(SA)_j + \sum_{k=1}^p C(SS)_k$$

Where

$C(M)_i$ = cost of module i

$C(SA)_j$ = cost of subassembly j; it is the sum of the cost of the modules it contains plus the cost of packaging the modules into a subassembly.

$C(SS)_k$ = cost of assembly k; it is the sum of the cost of the subassemblies plus the cost of packaging the subassemblies in an assembly.

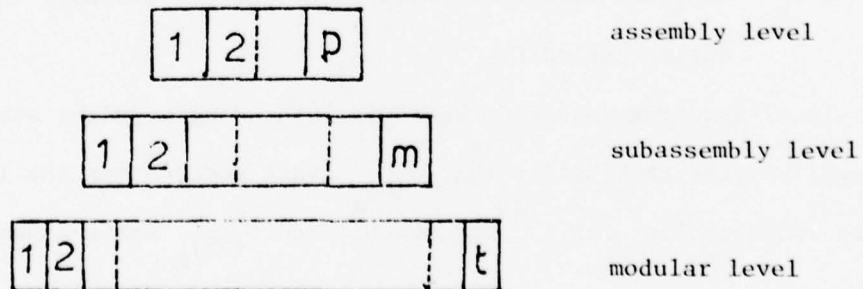


Figure 6: A multilevel design, D.

There are primarily two approaches to the problem of spares allocation to the multilevel designs:

- (a) Given the overall system availability, determine a spares allocation policy simultaneously for all levels.
- (b) Given the overall system availability, determine a spares policy for each level separately after the modularization process, by imposing the availability constraint at each level.

Consider (a); this procedure to find a spares allocation system is more accurate, but the equipment must be modularized into all levels before the spares calculation can be done. Using (b) is a more conservative approach, that is it requires that the system availability requirement be met at each level. Under model (a) the expression for the spares requirement follows.

Let $\theta(M)_i$ = expected number of failures of the i^{th} module in the design during period L.

$\theta(SA)_j$ = expected number of failures of the j^{th} subassembly in the design during period L.

$\theta(SS)_k$ = expected number of failures of the k^{th} assembly in the design during period L.

Thus, assuming independence of failure among modules, subassemblies, and assemblies, the probability that $S_{(M)_i}^*$ spare modules for the i^{th} module, $S_{(SA)_j}^*$ spare subassemblies for j^{th} subassembly and $S_{(SS)_k}^*$ spare assemblies for k^{th} assembly is sufficient over the operational life of the equipment is given by:

$$P_{(M)_i} = P(W(M)_i \leq S^*(M)_i) = \frac{\sum_{W(M)_i=0}^{S^*(M)_i} e^{-\theta(M)_i} [\theta(M)_i]^{W(M)_i}}{W(M)_i!}$$

$$P_{(SA)_j} = P(W(SA)_j \leq S^*(SA)_j) = \frac{\sum_{W(SA)_j=0}^{S^*(SA)_j} e^{-\theta(SA)_j} [\theta(SA)_j]^{W(SA)_j}}{W(SA)_j!}$$

$$P_{(SS)_k} = P(W(SS)_k \leq S^*(SS)_k) = \frac{\sum_{W(SS)_k=0}^{S^*(SS)_k} e^{-\theta(SS)_k} [\theta(SS)_k]^{W(SS)_k}}{W(SS)_k!}$$

Where, $W(M)_i$, $W(SA)_j$, $W(SS)_k$ are the total number of failures for the i^{th} module, the j^{th} subassembly, and the k^{th} assembly. Hence if the system availability is AV then it is required that:

$$\prod_{i=1}^t P_{(M)_i} \prod_{j=1}^m P_{(SA)_j} \prod_{k=1}^p P_{(SS)_k} \geq AV$$

The problem of optimal spares allocation becomes three dimensional and the number of possible elements whose spares can be determined at each iteration is $t \times m \times p$, making the problem increasingly difficult. Under model (b) the availability constraint is imposed at each level, thus making the constraint tighter. But as independence between the levels is also assumed, it is possible to arrive at a spares allocation policy without a significant shift from the results of method (a). Under (b) it is required

$$\prod_{i=1}^t P_{(M)_i} \geq AV$$

$$\prod_{j=1}^m P_{(SA)_j} \geq AV$$

$$\prod_{k=1}^p P_{(SS)_k} \geq AV$$

A separate policy for each level can be determined once the failure rate of each element in each level is determined.

Let $S^*(M)_i$, $S^*(SA)_j$ and $S^*(SS)_k$ denote the spares needed for the i^{th} module, the j^{th} subassembly, and the k^{th} assembly respectively. The cost of this spares policy is:

$$C_S = \sum_{i=1}^t C(M)_i S^*(M)_i + \sum_{j=1}^m C(SA)_j S^*(SA)_j + \sum_{k=1}^p C(SS)_k S^*(SA)_k$$

An expression for the life cycle cost (LCC) for a single equipment is:

$$\begin{aligned} \text{LCC} &= \sum_{i=1}^t C(M)_i + \sum_{j=1}^m C(SA)_j + \sum_{k=1}^p C(SS)_k \\ &+ \sum_{i=1}^t C(M)_i S^*(M)_i + \sum_{j=1}^m C(SA)_j S^*(SA)_j + \sum_{k=1}^p C(SS)_k S^*(SA)_k \end{aligned}$$

+ Inventory cost for each level

+ Cost of introducing a line item into inventory at each level.

IV. COMPUTER PROGRAM TO MODULARIZE LARGE NETWORKS

In the Multics environment the main program and all associated subroutines and functions are stored in separate segments. The following segments were created for the computer program developed.

Main. Prog. Fortran

Init. Fortran

Sort. Fortran

Sort. 1. Fortran

Mttr. Fortran

lcc. Fortran

Nspare. Fortran

Maxim. Fortran

M-shift. Fortran

R-shift. Fortran

Minx. Fortran

The above segments constitute the complete program. A brief description of the subroutines follows.

Main. prog. -- contains the coding for the modularization algorithm.

A. Subroutine Init:

Subroutine Init forms the $n \times n$ interconnection matrix from the network data.

B. Subroutines Sort and Sort 1:

Subroutines Sort and Sort 1 arrange the identification of the elements in each module.

C. Subroutine Mttr.:

Subroutine Mttr. evaluates the mean time to repair for a selected design. As the network grows in size and complexity, the mean time to repair which is a factor dependent on the interconnections between modules

as well as the number of modules becomes an important factor. The modularization process incorporates a methodology for evaluating this factor. The expression for the expected maintenance time is

$$E(TM) = T_1 + nT_2 + \sum_{i=1}^n T_3 \exp(T_4 P(M_i))$$

Where n = number of modules in a specific design

T_1 = a constant time per maintenance action

T_2 = a constant time per module

T_3 = a constant modifying the exponential relationship of the number of external connections

T_4 = a constant modifying the number of external connections

$P(M_i)$ = number of external connections to module i .

The specific form of the above equation as used by Caponecchi [3] is

$$E(TM) = 2.5 + .05n + .087 \sum_{i=1}^n \exp(.047P(M_i))$$

The above expression is evaluated for each design and the design is accepted if the calculated value of $E(TM) < MTTR \text{ max.}$

D. Subroutine Lcc:

This subroutine evaluates the life cycle cost (LCC) of the feasible designs. The main components of Lcc are the cost of acquisition and the support cost for the equipment over its intended useful life.

$$Lcc = C_A + C_S$$

Where C_A = acquisition cost for design

C_S = support cost for the design

Consider the acquisition cost C_A , which can be further broken down into

$$C_A = NE \left(\sum_{i=1}^n C(M_i) \right) \quad (1)$$

NE = number of equipments to be procured.

$C(M_i)$ = cost of each module in a design.

The cost of a module is further given as

$$C(M_i) = CC(NP) + CE(P(M_i)) + CP$$

Where CC = cost of a component

NP = number of components

CE = cost of providing external connections

$P(M_i)$ = number of external pin for module i

CP = cost of packaging a single module.

Under the assumptions of a discard at failure maintenance (DAFM) policy and that at least one of each module will be spared, the life time support cost of NE equipments, if each is at a separate site is:

$$CS = n CI + n CCL + NE \cdot \sum_{i=1}^n N_i \cdot C(M_i) \quad *$$

CS = Total organization support cost

CI = Cost of introducing a line item into the inventory system.

CC = Cost of maintaining a line item in inventory for one year.

L = Planned operational life of the equipment in years.

N_i = Number of spares of module i to be procured to support each equipment.

Then the total life cycle cost is the sum of equations (1) and (2) or

$$LCC = n(CI + CCL) + NE \sum_{i=1}^n (1 + N_i) C(M_i).$$

This cost is evaluated for each feasible design by subroutine Lcc.

A flow chart for subroutine Lcc is presented as Figure 7.

*Assumes all modules in equipment different.

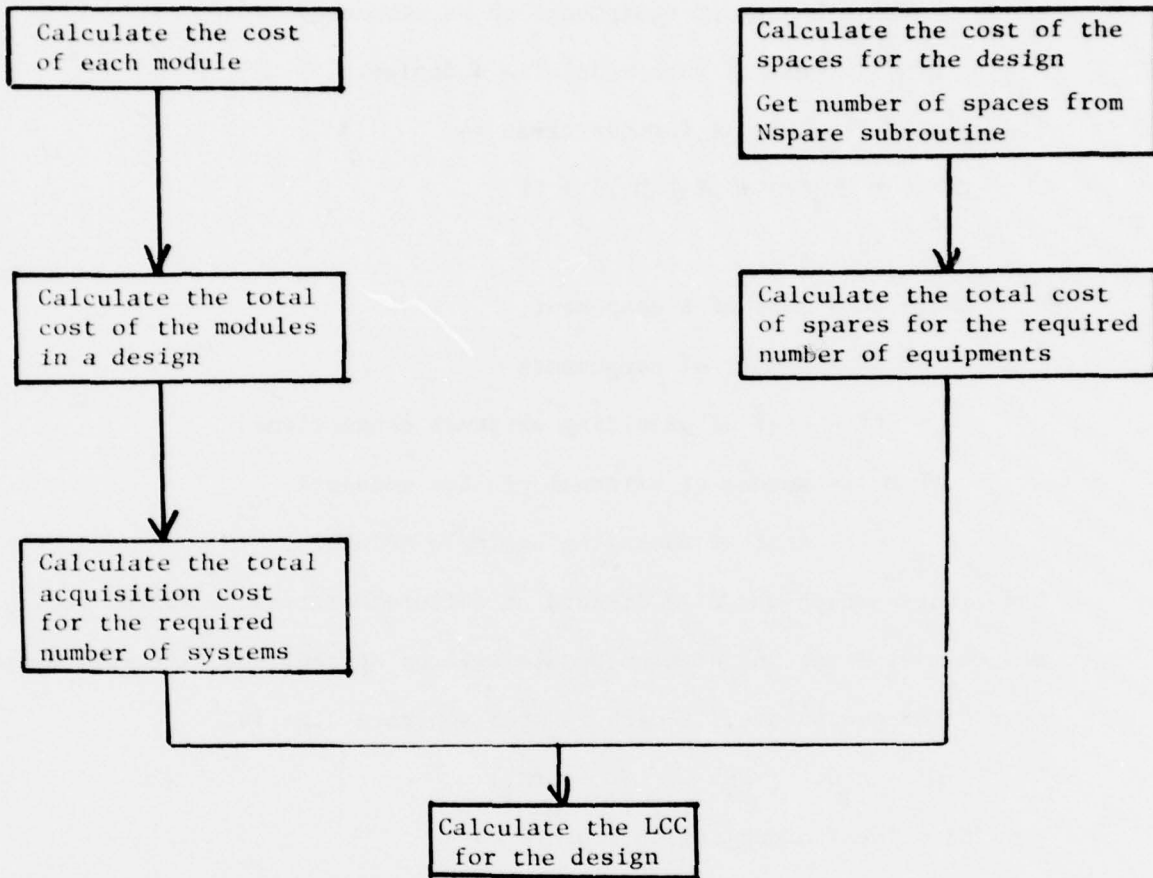


Figure 7: Flow Chart for LCC Subroutine

E. Subroutine Nspare:

A complete evaluation of the spares requirement is done by this subroutine in an iterative manner. The modules formed are treated as a series subsystem. Thus, if the system availability is A then the probability that N_i spares for module i is sufficient for the operational life of the system is given by

$$P(W_i \leq N_i) = \sum_{W_i=0}^{N_i} e^{-F_i} \frac{F_i^{W_i}}{W_i!} \geq A_{\min}$$

where $F_i = r_i \cdot L \cdot NE$, and

$$r_i = \left(\prod_{j \in M_i} r_j \right) \cdot r_p$$

r_i = failure rate of module i.

r_p = interconnecting pin failure rate reduction factor.

$AP(M_i)$ = The number of external connections eliminated by forming module i.

L = operational life of equipment.

NE = number of equipments.

But this inequality must be maintained for all modules in the system. Thus

$$\prod_{i=1}^n \sum_{W_i=0}^{N_i} e^{-F_i} \frac{F_i^{W_i}}{W_i!} \geq A$$

where n is total number of modules generated.

Therefore, an optimal spares policy is a set of $[N_1^* \dots N_n^*]$ such that:

$C_1 N_1 + C_2 N_2 \dots + C_n N_n$ is minimum where C_i is the cost of module i.

Or minimize:

$$C_1 N_1 + \dots + C_n N_n$$

Subject to:

$$\prod_{i=1}^n \sum_{W_i=0}^{N_i} e^{-F_i} \frac{F_i^{W_i}}{W_i!} \geq A$$

$$N_1, N_2, \dots, N_n \geq 0 \text{ and integer}$$

The above problem is an integer, non-linear programming problem, and is difficult to solve even for small values of n . But for large networks the number of modules generated, N , is fairly large and makes the problem increasingly difficult with increases in N . The alternative approximate solution method is similar to that in Biegel and Bulcha [1,2].

The solution method is:

1. Initialize by finding a lower bound N_i for each module, choose

N_i such that

$$\sum_{W_i=0}^{N_i} e^{-F_i} \frac{F_i^{W_i}}{W_i!} \geq A^*$$

2. Find the value of A for the system

$$\text{where } A = \prod_{i=1}^n \left(\sum_{W=0}^n e^{-F_i} \frac{F_i^W}{W!} \right)$$

If $A \geq A^*$ stop.

3. Let $N_i = N_i + 1$

$$4. \text{ Calculate } \Delta A_i^* = \left(\sum_{W=0}^{N_i+1} e^{-F_i} \frac{F_i^W}{W!} \right) - \left(\sum_{W=0}^{N_i} e^{-F_i} \frac{F_i^W}{W!} \right)$$

$$5. \text{ Calculate } \frac{\Delta A_i^*}{C(M_i)}$$

$$6. \text{ Choose } \max \frac{\Delta A_i^*}{C(M_i)} \text{ and set all other } N_i = N_i - 1$$

7. Go to step 2.

Now A^* is the minimum availability that each module must have for the overall availability of the equipment to be greater or equal to specified availability A . Theoretically $A^* = A^{1/n}$. When n is large A^* approaches 1, such that the above procedure overestimates the optimal spares policy. A heuristic procedure to avoid this problem was to set $A^* = A^{2/n}$. For instance if $A = .85$ and $n = 25$

$$A_{11}^* = A^{1/n} = (.85)^{1/25} \rightarrow A_{11} = .9935, A = .850$$

$$A_{21}^* = A^{2/n} = (.85)^{2/25} \rightarrow A_{21} = .9870, A = .721$$

Thus A_{21}^* is a better initial policy for the optimizing procedure.

The above procedure is coded into fortran and stored in subroutine N spare. A flow chart for subroutine N spare is presented as Figure 8.

F. Subroutine Maxim:

Subroutine maxim is used in subroutine N spare to select the modules with maximum availability/cost ratio.

G. Subroutine M-shift:

Subroutine M-shift contains a subroutine to rearrange the data matrix so that every node is considered as a starting node for a set of modules.

H. Subroutine R-shift:

Subroutine R-shift contains a subroutine to reassign physical property values after the matrix has been rearranged by M-shift.

I. Function Minx:

A function used by the main program for selecting the group of elements with the minimum value for the external minus the internal connections at each iteration.

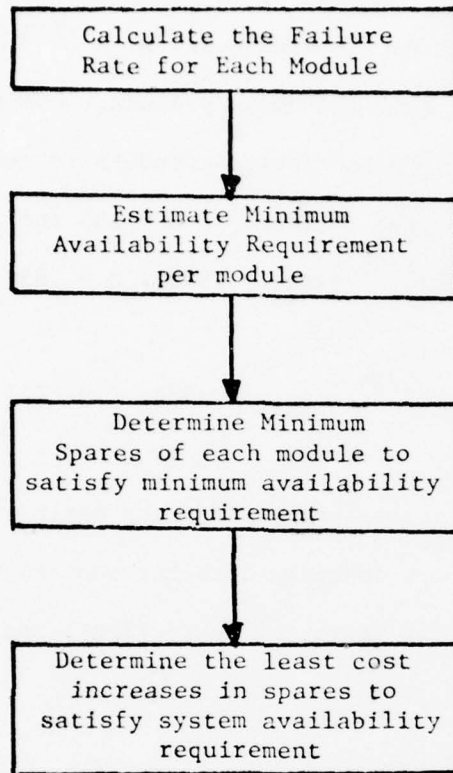


Figure 8: Flow Chart for Nspare Subroutine.

The following data files contain the data that are required by the different segments.

- data-2 contains the maximum maintenance time allowed.
- data-5 contains the numbers of nodes n , physical data for nodes and the set of physical constraints.
- data-7 contains the number of equipments to be procured, the cost of external pins and the shelf cost of maintaining a module.
- data-8 (1) The maximum length from a diagonal element up to and including the last non-zero element in the matrix representation of the network.
(2) row number and row entries starting from the diagonal up to and including the last non-zero element in the row.
- data-9 availability figure for the system.
- data-10 node failure rates (enter only decimal figures, eg. .211)

An example of a 100 node network that was modularized is included. The data for the network was generated in a random manner and stored in the appropriate files.

Summary:

This research has developed a method for modularizing large networks subject to physical constraints, mean time to repair constraints and availability constraints. A procedure has also been developed for the spares allocation and life cycle cost evaluation of the modularization designs.

A solution methodology for any higher level assembly is introduced by repeating the modularization procedure with the appropriate modification of the lower level designs, which will facilitate the problem as no additional algorithm is needed.

REFERENCES

- [1] Biegel, John E. and Bisrat Bulcha, System Modularization to Minimize Life Cycle Costs, Technical Report, July 1976, Department of Industrial Engineering and Operations Research, College of Engineering, Syracuse University, Syracuse, NY 13210.

- [2] Biegel, John E. and Bisrat Bulcha, System Modularization to Minimize Life Cycle Costs, RADC-TR-77-13, January 1977, A036868, Rome Air Development Center, Air Force Systems Command, Griffiss Air Force Base, Rome, New York 13441.

- [3] Caponecchi, A. J., A Methodology For Obtaining Solutions to the Modular Design Problem, Ph.D. Dissertation, The University of Texas at Austin, 1971.

APPENDIX 1

Failure rates for modules is addressed in [1,2] but for convenience it will be repeated below.

Let r_i denote failure rate for module i .

$$\text{Then } r_i = \sum_{j \in M_i} r_j - \Delta P(M_i) r_p \quad (\text{A1-1})$$

where r_j = the failure rate of element j of module i

$\Delta P(M_i)$ = the number of external connections eliminated by putting elements j in module i

r_p = failure rate correction factor for interconnection reduction.

$$\text{Then } \theta(M)_i = r \cdot L \cdot NE$$

Similarly to calculate failure rate of a subassembly substitute for the appropriate module failure rates r_j in (1), and the number of ΔP for external pins eliminated by combining the modules into a subassembly. Also for an assembly substitute the appropriate failure rate for the subassemblies for r_j in (1) and the number of external pins eliminated by combining subassemblies into an assembly ΔP .

APPENDIX 2: THE MODULARIZATION PROGRAM

PP main.fortran

main.fortran 11/28/77 1048.2 est Mon

```

common tmax,av,co,ce,cr,ne,ci,cc,lit
dimension rj(100),r(100),rnt(100),rlcc(100)
dimension nst(100)
dimension c1(100),c2(100),c3(100)
dimension bndrs(3),i1(100),e1(100)
integer a(100,100),m(110,110),s(100),st(100),sd(100)
integer mv(100),mvt(100),t(100)
integer cno,ext
integer co,ce,cr,ci,cc
integer trow(100)
integer n1(100),i11(100),i12(100)
data bndrs/150.,1f00.,0.17/
data a/10000%0/
data i11/100%0/,i12/100%0/,ifeas/0/,vnr/100%0/
data rnt/100%0./,r/100%0./,rlcc/100%0./
data t1cc/0./
read(9,20) av
read(7,10) co,ce,cr,ne,ci,cc,lit
read(2,11)tmax
20 format(f5,3)
10 format(7i5)
11 format(f6.0)
read(10,922) (rj(j),j=1,100)
922 format(8f10.5)
921 format(i3)
cno=0
c fetch connections matrix
read(1,12) n
12 format(i5)
write(3,13) n,n
13 format(' network partitioning program ',//,' input matrix '
,'should be ',i3,' by ',i3)
read(1,21)(c1(i),i=1,n)
read(1,21)(c2(i),i=1,n)
read(1,21)(c3(i),i=1,n)
21 format(8f10.0)
call init(n,2)
do 9999 loop=1,n
write(3,9995)loop
9995 format('0 starting node:',i3)
ls=n
do 110 i=1,n
s(i)=1
t(i)=0
do 110 j=1,n
110 t(i)=t(i)+a(i,j)
1000 if(ls .eq. 0)go to 5000
mvt(1)=min0(s,ls)
lmt=1
go to 2020
2000 j1=i.t(imin)
j2=ext(imin)

```

THIS PAGE IS BEST QUALITY PRACTICABLE
FROM COPY FURNISHED TO DDC

```

      JS=st(1mvt)
2020 call sort(mvt,mv,lmvt)
      lmv=lmvt
      lsd=0
      do 300 i=1,ls
      do 280 j=1,lmv
      if(s(i).eq.mv(j))go to 300
280 continue
      lsd=lsd+1
      sd(lsd)=s(i)
300 continue
      if(lsd .eq. 0)go to 4000
      do 500 i=1,lsd
      do 400 j=1,lmv
400 m(i,j)=mv(j)
500 m(i,lmv+1)=sd(i)
      lmv=lmv+1
      do 600 i=1,lsd
      int(i)=0
      do 600 j=1,lmv
      l=j+1
      do 600 k=1,lmvt
600 int(i)=int(i)+z(m(i,j)*m(i,k))
      do 700 i=1,lsd
      ext(i)=0
      do 750 j=1,lmvt
750 ext(i)=ext(i)+t(m(i,j))
      ext(i)=ext(i)-2*int(i)
700 st(i)=ext(i)-int(i)
      imin=1
      low=st(imin)
      do 800 i=1,lsd
      if(st(i) .se. low) go to 800
      imin=i
      low=st(imin)
800 continue
      con1=0.
      con2=0.
      con3=0.
      do 900 j=1,lmvt
      mvt(j)=m(imin,j)
      con1=con1+t1(mvt(j))
      con2=con2+t2(mvt(j))
      con3=con3+t3(mvt(j))
900 if((con1 .le. bndry(1)) .and. (con2 .le. bndry(2)) .and.
      (con3 .le. bndry(3))) go to 2000
      lst=0
      do 980 i=1,ls
      do 950 j=1,lmv
      if(s(i).eq.mv(j)) go to 980
950 continue
      lst=lst+1
      st(lst)=s(i)
980 continue
      ls=lst

```

```

do 990 i=1,lc
990 s(i)=st(i)
4000 cno=cno+1
call sort(mv,st,lmv)
do 901 i=1,lmv
nst(i)=st(i)+(loop-1)
if(nst(i).gt.n)nst(i)=nst(i)-n
901 continue
call sort1(nst,lmv)
write(3,180)cno,(nst(i),i=1,lmv)
180 format('0 card no. ',i3,' contains nodes:',/,',0',20i5,100(/,6x,
119i5))
do 91 i=1,lmv
nr(cno)=nr(cno)+c1(nst(i))
91 continue
do 92 i=1,lmv
rnt(cno)=rnt(cno)+rj(nst(i))
92 continue
r(cno)=rnt(cno)
write(3,190)j2,j1,j3
190 format('0',//,' ext. conn. = ',i5,/' int. conn. = ',i5,/, ' st = ',
i5)
ij1(cno)=j1
ij2(cno)=j2
if(lsd.ne. 0) go to 1000
write(3,195)
195 format('0 * * * * *_r u n c o m p l e t e d * * * * * ')
call mtr(cno,ij2,ifeas)
call lcc(ifeas,nr,ij2,cno,ij1,r,tlcc)
rlcc(loop)=tlcc
call rshift(c1,n)
call rshift(c2,n)
call rshift(c3,n)
call mshift(a,n)
do 777 j=1,cno
rnt(j)=0
nr(j)=0
ij1(j)=0
ij2(j)=0
777 continue
cno=0
9999 continue
stop
5000 write(3,195)
stop
end

```

r 1049 0.808 0.710 51

THIS PAGE IS BEST QUALITY PRACTICABLE
FROM COPY FURNISHED TO DDG

THIS PAGE IS BEST QUALITY PRACTICABLE
FROM COPY FURNISHED TO DDG

PR init.fortran

init.fortran

11/28/77 1100.1 est Mon

```
subroutine init(na,aa)
integer aa(na,na),col(100)
read(8,49)n
49 format(i2)
39 read(8,29,END=99) i,(col(j),j=1,n)
29 format(i3,30i2)
do 19 j=1,n
  l=i+j
  if (l.st.100) go to 19
  aa(i,l)=col(j)
19 continue
  go to 39
99 do 10 i=1,na
  do 15 j=1,na
  if (j+i.st.100) go to 15
  aa(i+j,i)=aa(i,j+i)
15 continue
10 continue
return
end
```

r 1100 0.093 0.246 17

PR sort.fortran

sort.fortran

11/28/77 1102.2 est Mon

```
subroutine sort(inp,outp,n)
integer inp(n),outp(n),bptr,bsub,tempe
c bubble sort.
c optimum sort conditions occur in a completely sorted array
c number of compares in a sorted array is n-1
  isent=1
  do 10 i=1,n
10 outp(i)=inp(i)
  if(n .le. 1) return
15 do 20 bptr=2,n
  bsub=bptr-1
  if(outp(bptr) .ge. outp(bsub)) go to 20
```

```

temp=outs(bstr)
outs(bstr)=outs(bsub)
outs(bsub)=temp
inscnt=inscnt+1
go to 15
20 continue
return
end

```

sort1.fortran

sort1.fortran 11/28/77 1101.6 est Mon

```

subroutine sort1 (outs,i,n)
integer outs(i(n)),bstr,temp,bsub
inscnt=1
if(n.le.1) return
15 do 20 bstr=2,n
    bsub=bstr-1
    if(outs(bstr).ge.outs(bsub)) go to 20
    temp=outs(bstr)
    outs(bstr)=outs(bsub)
    outs(bsub)=temp
    inscnt=inscnt+1
20 go to 15
continue
return
end

```

pr mtr.fortran

mtr.fortran 11/28/77 1051.6 est Mon

```

subroutine mtr(cno,iJ2,ifeas)
common tmax,av,co,ce,cp,ne,ci,cc,lit
integer co,ce,cp,ci,cc
integer cno
integer iJ2(cno)
n=0
do 10 i=1,cno
    xm=xm*exp(.047*iJ2(i))
10 continue
xm=xm*.087
ym=2.5+.05*cno
etm=xm+ym
13 write (3,13) etm
    format(2x,'expected total maintenance time=',f8.5)
    if (etm.le.tmax) go to 20
write (3,12)
12 format(2x,'design not feasible')
ifeas=-1

```

THIS PAGE IS BEST QUALITY PRACTICABLE
FROM COPY FURNISHED TO DDG

```
20 return
   ifeas=1
return
end
```

THIS PAGE IS BEST QUALITY PRACTICABLE
FROM COPY FURNISHED TO DDC

pr lcc.fortran

lcc.fortran

11/28/77 1054.6 est Men

```
subroutine lcc(ifeas,ne,iJ2,cno,iJ1,r,tlcc)
common tmax,av,co,ce,cs,ne,ci,cc,lit
integer co,ce,cs,ci,cc
integer tcm,tesp
integer cno
integer csf,tlcc
integer ac,sc
dimension ne(cno),iJ2(cno), iJ1(cno),r(cno)
dimension cm(100),ns(100)
data ns/100*0/
data cm/100*0./
if (ifeas.eq.-1) go to 40
tcm=0
do 15 i=1,cno
cm(i)=(co*ne(i))+(ce*iJ2(i))+cs
15 continue
do 20 i=1,cno
write (3,51) i,cm(i)
51 format(5x,'cost of mod.',i3,'=',f10.3)
tcm=tcm+cm(i)
20 continue
write(3,52) tcm
52 format(2x,'total cost of the modules=',i6)
ac=ne*tcm
write (3,88)ac
88 format(2x,'acquisition cost=',i10)
call nspare(ns,cno,cm,iJ1,r)
tesp=0
do 25 i=1,cno
tesp=tesp+(ns(i)*cm(i))
25 continue
csf=ne*tesp
write (3,48) csf
48 format(2x,'total cost for the spares requirement=',i10)
sc=cno*ci+cno*cc+lit+tesp
write (3,99)sc
99 format(2x,'support cost=',i10)
```

```

      tlec=se+ge
      write (3,30) tlec
30      format(2x,'the cost of this design is-',i10)
      return
40      tlec=10**6
      return
      end

```

r 1054 0.454 0.528 31

PR nspare.fortran

nspare.fortran 11/28/77 1052.6 est Mon

```

subroutine nspare(ns,cno,cm,i,j1,r)
common tmax,av,co,ce,cf,ne,ci,cc,lit
integer co,ce,cf,ci,cc
integer cno
integer ns(cno),i,j1(cno)
dimension pr(100)
dimension r(cno),cm(cno)
dimension f(100),tempx(100),dlays(100),temex1(100),sel(100)
data f/100*0./
fnewv=365.*24.
ft=fnewv*(10.**(-5))
write (3,99) ne
99      format(2x,'no. of equipments to be procured=',i5)
write (3,209) lit
209     format(2x,'expected no of failures in ',i2,'years')
do 10 i=1,cno
f(i)=ft*r(i)*lit*ne
write (3,80) i, f(i)
80      format(2x,'mod.',i3,'=',f7.3)
10      continue
70      avs=1.
      avmin=(av**(2./cno))
do 25 j=1,cno
tempx(j)=0.
ix=0
pr(j)=exp(-f(j))
35      tempx(j)=tempx(j)+pr(j)
      if (tempx(j).ge.avmin) go to 30
      ix=ix+1
pr(j)=pr(j)*f(j)/ix
go to 35
30      ns(j)=ix
      avs=tempx(j)*avs
25      continue
if (avs.ge.av) go to 40
110     do 50 j=1,cno
dlays(j)=pr(j)*f(j)/(ns(j)+1)

```

THIS PAGE IS BEST QUALITY PRACTICABLE
FROM COPY FURNISHED TO DDG

```

50   sel(j)=dlays(j)/cm(j)
      continue
      call maxim (sel,cno,k)
      temex(k)=temex(k)+dlays(k)
      ns(k)=ns(k)+1
      avs=1.
      do 100 j=1,cno
100   avs=temex(j)*avs
      continue
      if(avs.ge.av) go to 40
      go to 110
40   write(3,45) (ns(j),j=1,cno)
45   format(2x,'spheres mix for this design',20(i3,2x))
      write (3,88) avs
88   format(2x,'system av.=',f8.5)
      return
      end

```

r 1052 0.346 0.072 8

PR maxim.fortran

maxim.fortran

11/28/77 1056.7 est Mon

```

      subroutine maxim(x,n,l)
      dimension x(n)
      tmax=x(1)
      l=1
      if(n.le.1) return
      do 10 i=2,n
      if(x(i).le.tmax) go to 10
      tmax=x(i)
      l=i
10   continue
      return
      end

```

PR rshift.fortran

rshift.fortran

11/28/77 1059.3 est Mon

```

      subroutine rshift(row,n)
      real row(n), temp
c   temp is a temporary storage for first element to be shifted
      temp=row(1)
      do 12 i=2,n
      row(i-1)=row(i)
12  continue
      row(n)=temp
      return
      end

```

THIS PAGE IS BEST QUALITY PRACTICABLE
FROM COPY FURNISHED TO DDC

PR mshift.fortran

mshift.fortran 11/28/77 1058.7 est Mon

```
subroutine mshift(mat,n)
integer mat(100,100),trow(100)
c   trow will store the first row to be shifted
do 113 i=1,n
113 trow(i)=mat(1,i)
do 212 i=2,n
do 212 j=1,n
212 mat(i-1,j)=mat(i,j)
do 213 i=1,n
213 mat(n,i)=trow(i)
c   to shift column
do 313 j=1,n
313 trow(j)=mat(j,1)
do 312 i=1,n
do 312 j=2,n
312 mat(i,j-1)=mat(i,j)
do 413 j=1,n
413 mat(j,n)=trow(j)
return
end
```

PR minx.fortran

minx.fortran 11/28/77 1056.3 est Mon

```
function minx(x,n)
integer x(n)
minx=x(1)
if(n .le. 1) return
do 10 i=2,n
10 if(x(i) .lt. minx) minx=x(i)
return
end
```

THIS PAGE IS BEST QUALITY PRACTICABLE
FROM COPY FURNISHED TO DDG

APPENDIX 3: EXAMPLE OF USE OF MODULARIZATION PROGRAM

The user would create the data files shown on the following four pages. All of the segments of the main program must be compiled and the data attached. The following shows the instructions required for attaching the data files by the segment run. ec.

PR run.ec

run.ec 11/29/77 1132.5 est Tue

```
io_call attach file01 vfile_ data_5
io_call attach file02 vfile_ data_2
io_call attach file07 vfile_ data_7
io_call attach file08 vfile_ data_8
io_call attach file09 vfile_ data_9
io_call attach file10 vfile_ data_10
```

main.prog

```
io_call detach file01
io_call detach file02
io_call detach file07
io_call detach file08
io_call detach file09
io_call detach file10
```

r 1132 0.252 0.932 42

THIS PAGE IS BEST QUALITY PRACTICABLE
FROM COPY FURNISHED TO DDC

Pr data_2

data_2 11/29/77 1136.4 est Tue

40.1

r 1136 0.044 0.506 22

r 1512 0.037 0.006 2

Pr data_7

data_7 10/02/77 1513.1 edt Sun

2 2 100 30 103 12 10

r 1513 0.039 0.002 1

Pr data_8

data_8 10/02/77 1513.2 edt Sun

30

1 6 7 2 2 6
2 5 0 0 0 8
3 7
4 0 0 2 3
5 3 0 2
6 0 2 4 0 0 0 8
7 0 0 3 2 7
8 0 0 0 4 5
9 0 0 0 6 5
10 5 0 0 0 8
11 0 0 0 9 5 3
12 0 0 0 0 9 1
13 0 0 0 0 6 4
14 0 0 0 0 2 0 0 0 6
15 3 0 0 0 5
16 0 0 0 3 4 0 0 0 4
17 7 0 0 7
18 0 0 4 0 0 0 0 5
19 0 5 1 0
20 0 0 0 8 6 0 0 0 5
21 4 0 0 7 1
22 1 0 0 0 7
23 0 0 0 0 8
24 0 0 0 0 1 5 0 0 1
25 0 0 0 0 4
26 9 0 0 2 3
27 4 0 0 0 8
28 0 0 6 9 0 0 3 0 8 5
29 2 0 0 7 0 0 3
30 0 0 0 9 0 0 0 0 8
31 7 0 2
32 0 0 4 0 7
33 5 0 4 0 0 0 0 0 8
34 0 0 9 0 2 3
35 0 6 1 0
36 0 0 9 0 0 0 7
37 4 0 5 7

THIS PAGE IS BEST QUALITY PRACTICABLE
FROM COPY FURNISHED TO DDC

pr data_5

data_5 10/02/77 1510.7 edt Sun

1.	100							
\c.	31.	34.	20.	13.	41.	41.	53.	28
\c.	34.	48.	12.	13.	34.	41.	11.	28
\c.	28.	14.	29.	41.	37.	52.	49.	34
\c.	15.	40.	29.	42.	51.	45.	22.	13
\c.	44.	25.	39.	45.	55.	27.	22.	55
\c.	43.	44.	40.	14.	39.	50.	23.	30
\c.	45.	32.	21.	23.	27.	18.	32.	51
\c.	13.	51.	33.	34.	55.	25.	33.	22
\c.	15.	53.	14.	33.	28.	23.	52.	14
\c.	31.	53.	13.	45.	45.	48.	16.	11
\c.	41.	50.	39.	44.	43.	55.	50.	21
\c.	24.	26.	34.	37.	49.	29.	48.	23
2.	29.	35.	32.	23.				
\c.	120.	114.	206.	257.	88.	198.	190.	291
\c.	245.	203.	276.	218.	266.	116.	127.	238
\c.	109.	101.	141.	81.	172.	86.	237.	287
\c.	133.	120.	150.	276.	224.	114.	230.	165
\c.	166.	190.	113.	210.	267.	210.	291.	203
\c.	113.	297.	170.	112.	205.	136.	188.	183
\c.	292.	108.	124.	151.	219.	108.	224.	217
\c.	257.	135.	185.	166.	125.	87.	279.	174
\c.	112.	289.	171.	109.	275.	101.	275.	101
\c.	116.	96.	161.	136.	188.	253.	181.	253
\c.	180.	258.	285.	224.	128.	230.	280.	136
\c.	270.	184.	192.	213.	260.	260.	247.	182
\c.	290.	220.	177.	262.				
	0.037	0.015	0.019	0.015	0.032	0.011	0.018	0.018
	0.016	0.033	0.049	0.025	4 0.044	0.049	0.021	0.029
	0.027	0.043	0.046	0.02	0.041	0.038	0.029	0.021
	0.028	0.014	0.038	0.026	0.044	0.015	0.049	0.012
	0.014	0.03	0.041	0.048	0.023	0.025	0.041	0.04
	0.03	0.037	0.038	0.041	0.012	0.012	0.019	0.033
	0.039	0.032	0.022	0.043	0.025	0.038	0.023	0.011
	0.015	0.031	0.042	0.019	0.013	0.013	0.017	0.039
	0.022	0.013	0.03	0.029	0.040	0.028	0.013	0.028
	0.044	0.038	0.034	0.038	0.034	0.024	0.013	0.017
	0.032	0.012	0.015	0.049	0.018	0.025	0.035	0.032
	0.012	0.045	0.026	0.03	0.034	0.04	0.038	0.032
	0.029	0.018	0.035	0.039				

r 1511 0.349 0.002 1

38 0 6 0 4 0 0 0 0 0 9
 39 5 0 0 0 4 9 0 0 0 6
 40 5 0 0 0 1 2
 41 3 0 0 0 7 6
 42 0 0 0 0 0 2
 43 0 0 0 0 0 3 0 8
 44 3 0 0 0 0 4
 45 3
 46 5 0 0 7 0 0 4 4 0 0 6 9
 47 0 0 0 0 0 0 2 0 0 0 8
 48 0 0 0 0 0 9 5
 49 2 0 2
 50 0 7 4
 51 4 0 0 0 4 0 0 0 0 2
 52 8 0 0 7
 53 0 0 0 7 0 0 0 0 0 0 0 0 5
 54 3 0 0 6 1 5
 55 0 0 5 0 6 0 0 0 0 1
 56 7 0 0 0 0 9 0 0 0 0 2
 57 7 0 0 8 4
 58 7 0 0 0 9 8
 59 3 0 0 0 2 7
 60 0 0 0 0 3
 61 0 0 0 0 4 4 0
 62 8 0 0 0 2 5
 63 4 0 0 0 2 8
 64 4 0 0 0 6 3
 65 0 0 0 0 6 0 0 0 0 0 0 5
 66 1 0 0 0 3 0 4
 67 2 0 0 8 7
 68 7 0 0 5
 69 0 0 0 0 0 2 3 9
 70 0 0 0 0 0 0 8
 71 0 0 2 9
 72 0 0 3 4
 73 5 0 0 0 0 4
 74 0 0 0 0 4 6
 75 0 0 0 0 8
 76 0 0 0 7 6 7 0 0 2
 77 2 0 0 0 1
 78 0 0 0 4
 79 0 0 0 6 0 0 0 0 5
 80 3 0 8 7 0 1
 81 0 0 0 4 3
 82 0 0 5 0 1
 83 0 0 0 0 9
 84 0 0 0 0 4 1
 85 1 0 0 0 6 2
 86 2 0 0 3
 87 0 0 0 7 0 0 0 0 5
 88 5 0 0 8 4
 89 6 0 4 0 9
 90 2 0 0 4 1 0 0 9
 91 0 0 0 6
 92 3 2
 93 0 0 0 8
 94 0 0 1 0 0 8
 95 0 0 6
 96 0 0 4
 97 0 0 3
 98 3 6
 99 5
 100 0

r 1513 0.248 0.002 1

pr data_9

data_9 10/02/77 1514.7 edt Sun

.85

r 1514 0.037 0.002 1

pr data_10

data_10 10/02/77 1515.0 edt Sun

.359	.322	.107	.341	.408	.181	.242	.364
.254	.168	.182	.213	.142	.261	.413	.417
.103	.415	.267	.272	.1999	.446	.263	.179
.252	.429	.099	.362	.365	.387	.127	.087
.335	.402	.313	.353	.349	.450	.409	.107
.194	.210	.270	.299	.39	.233	.392	.180
.278	.162	.098	.332	.332	.331	.246	.222
.273	.388	.093	.100	.276	.329	.083	.222
.278	.162	.098	.332	.331	.426	.222	.331
.105	.235	.335	.298	.425	.394	.275	.115
.322	.234	.340	.417	.363	.179	.098	.353
.202	.315	.360	.447	.216	.172	.444	.348
.146	.187	.254	.279				

r 1515 0.127 0.006 2

The computer output for the network described by the data files is shown in the pages following those files. The first 20 lines are the login procedure where.

Lines 1-9 ---standard login procedure

Line 10 ----asks for the output device to be appended

Line 12 ----type ec run to append input devices containing
data files, run the program and then detach input
devices.

Line 20 ----begins the computation, to find the design starting
from node 1.

(1) login kernel

File name: 11

XXXXXXXXXXXX

You are protected from screen 1 and 11 1135.

Binary 2567c0016 loaded in 1135 1135 1135 from 2567c0016 terminal "none"
Last login: 11/28/77 1127.9 out from 2567c0016 terminal "none".

No mail.

io.call: Improper mode specification for the device, user_i/o

r 1130 1.562 13.319 115

(10) io attach file03 ssm_ user_i/o

r 1130 0.120 0.262 6

(12) cc run

io.call attach file01 vfile_ data_5

io.call attach file02 vfile_ data_2

io.call attach file07 vfile_ data_7

io.call attach file08 vfile_ data_8

io.call attach file09 vfile_ data_9

io.call attach file10 vfile_ data_10

main.Prod

(2c) network partitioning program

input matrix: should be 100 by 100

0 starting node: 1

0 card no. 1 contains nodes:

0 1 2 3 4

0

ext. conn. = 21

int. conn. = 27

st = -6

0 card no. 2 contains nodes:

0 5 42 96

0

ext. conn. = 25

int. conn. = 0

st = 25

0 card no. 3 contains nodes:

0 6 8 9 13

0

ext. conn. = 33

int. conn. = 25

st = 8

0 card no. 4 contains nodes:

0 7 12 17 18 21

0

ext. conn. = 54

int. conn. = 35

st = 19

0 card no. 5 contains nodes:

0 10 11 15 16 20

0

ext. conn. = 35

int. conn. = 38

st = -3

0 card no. 6 contains nodes:

0 14 19 23

-

ext. conn. = 24

int. conn. = 0

st = 16

0 card no. 7 contains nodes:
0 22 81 86
0

ext. conn. = 33
int. conn. = 3
st = 30

0 card no. 8 contains nodes:
0 24 29 33
0

ext. conn. = 35
int. conn. = 9
st = 26

0 card no. 9 contains nodes:
0 25 39 44 45 49
0

ext. conn. = 60
int. conn. = 22
st = 38

0 card no. 10 contains nodes:
0 26 27 28 31 32
0

ext. conn. = 52
int. conn. = 46
st = 6

0 card no. 11 contains nodes:
0 30 34 78
0

ext. conn. = 58
int. conn. = 9
st = 49

0 card no. 12 contains nodes:
0 35 66 73
0

ext. conn. = 31
int. conn. = 4
st = 27

0 card no. 13 contains nodes:
0 36 43 51
0

ext. conn. = 37
int. conn. = 15
st = 22

0 card no. 14 contains nodes:
0 37 41
0

ext. conn. = 60
int. conn. = 7
st = 53

0 card no. 15 contains nodes:
0 38 48 54 55 60
0

ext. conn. = 47
int. conn. = 37
st = 10

THIS PAGE IS BEST QUALITY PRACTICABLE
FROM COPY FURNISHED TO DDC

0 card no. 16 contains nodes:
0 41 42 43
0

ext. conn. = 51
int. conn. = 9
st = 51

0 card no. 17 contains nodes:
0 46 50 52 53 57
0

ext. conn. = 80
int. conn. = 43
st = 37

0 card no. 18 contains nodes:
0 47 98 99 100
0

ext. conn. = 51
int. conn. = 14
st = 37

0 card no. 19 contains nodes:
0 56 62 63 69
0

ext. conn. = 61
int. conn. = 24
st = 37

0 card no. 20 contains nodes:
0 58 59 64 65 70
0

ext. conn. = 75
int. conn. = 37
st = 38

0 card no. 21 contains nodes:
0 61 91 95
0

ext. conn. = 36
int. conn. = 6
st = 30

0 card no. 22 contains nodes:
0 67 71 72 74 75
0

ext. conn. = 53
int. conn. = 29
st = 24

0 card no. 23 contains nodes:
0 69 77 82
0

ext. conn. = 53
int. conn. = 10
st = 43

0 card no. 24 contains nodes:
0 76 85 89 90
0

ext. conn. = 81
int. conn. = 14
st = 67

0 card no. 25 contains nodes:
0 79 83 88 92
0

ext. conn. = 34
int. conn. = 28
st = 6

THIS PAGE IS BEST QUALITY PRACTICABLE
FROM COPY FURNISHED TO DDC

0 card no. 26 contains nodes:
0 80 87 93 94
0

ext. conn. = 94
int. conn. = 0
st = 94

0 * * * * * r u n c o m p l e t e d * * * * *
expected total maintenance time=38.58499

cost of mod. 1=	338.000
cost of mod. 2=	366.000
cost of mod. 3=	440.000
cost of mod. 4=	498.000
cost of mod. 5=	450.000
cost of mod. 6=	386.000
cost of mod. 7=	462.000
cost of mod. 8=	428.000
cost of mod. 9=	490.000
cost of mod. 10=	496.000
cost of mod. 11=	452.000
cost of mod. 12=	408.000
cost of mod. 13=	386.000
cost of mod. 14=	416.000
cost of mod. 15=	476.000
cost of mod. 16=	458.000
cost of mod. 17=	550.000
cost of mod. 18=	428.000
cost of mod. 19=	506.000
cost of mod. 20=	538.000
cost of mod. 21=	446.000
cost of mod. 22=	498.000
cost of mod. 23=	452.000
cost of mod. 24=	538.000
cost of mod. 25=	394.000
cost of mod. 26=	566.000

total cost of the modules= 11866

acquisition cost= 355980

no. of equipments to be procured= 30

expected no of failures in 10years

mod. 1=	29.670
mod. 2=	25.386
mod. 3=	24.729
mod. 4=	30.824
mod. 5=	38.159
mod. 6=	20.787
mod. 7=	24.887
mod. 8=	23.100
mod. 9=	42.784
mod. 10=	29.013
mod. 11=	31.089
mod. 12=	15.242
mod. 13=	18.948
mod. 14=	14.270
mod. 15=	34.348
mod. 16=	17.608
mod. 17=	35.005
mod. 18=	29.223
mod. 19=	25.386
mod. 20=	36.276
mod. 21=	28.382
--	---

THIS PAGE IS BEST QUALITY PRACTICABLE
FROM COPY FURNISHED TO DDG

THIS PAGE IS BEST QUALITY PRACTICABLE
FROM COPY FURNISHED TO DDC

mod. 22: 42,094
mod. 23: 26,017
mod. 24: 30,976
mod. 25: 37,184
mod. 26: 15,794

Spares min for this design 43 45 37 44 53 32 37 35 38 41
44 25 37 30 48 28 49 42 37 51

41 45 38 44 52 25

system av. = 0.85257

total cost for the spares requirement = 14595420

support cost = 14401218

the cost of this design is = 14957198

0 starting node: 2

0 card no. 1 contains nodes:

0 2 7 12 17 18

0

ext. conn. = 47

int. conn. = 32

st = 15

0 card no. 2 contains nodes:

0 1 3 4 5 6

0

ext. conn. = 32

int. conn. = 27

st = 5

0 card no. 3 contains nodes:

0 8 42 87 96

0

ext. conn. = 39

int. conn. = 5

st = 34

0 card no. 4 contains nodes:

0 9 14 23

0

ext. conn. = 21

int. conn. = 11

st = 10

0 card no. 5 contains nodes:

0 10 11 15 16 20

0

ext. conn. = 35

int. conn. = 38

st = -3

0 card no. 6 contains nodes:

0 13 19 22

0

ext. conn. = 44

int. conn. = 5

st = 39

0 card no. 7 contains nodes:

0 21 25 81 86

0

ext. conn. = 59

int. conn. = 10

st = 49

0 card no. 8 contains nodes:

0 24 29 33

0

ext. conn. = 35

int. conn. = 9

st = 26

0 card no. 9 contains nodes:
0 26 27 28 31 32
0

ext. conn. = 52
int. conn. = 46
st = 6
0 card no. 10 contains nodes:
0 30 34 39 44 45
0

ext. conn. = 65
int. conn. = 35
st = 30
0 card no. 11 contains nodes:
0 35 78 82
0

ext. conn. = 35
int. conn. = 4
st = 31
0 card no. 12 contains nodes:
0 36 43 51
0

ext. conn. = 37
int. conn. = 15
st = 22
0 card no. 13 contains nodes:
0 37 41
0

ext. conn. = 60
int. conn. = 7
st = 53
0 card no. 14 contains nodes:
0 38 48 54 55 60
0

ext. conn. = 47
int. conn. = 37
st = 10
0 card no. 15 contains nodes:
0 40 66 73
0

ext. conn. = 44
int. conn. = 4
st = 40
0 card no. 16 contains nodes:
0 46 50 52 53 57
0

ext. conn. = 80
int. conn. = 43
st = 37
0 card no. 17 contains nodes:
0 47 84 93 97
0

ext. conn. = 44
int. conn. = 8
st = 36

THIS PAGE IS BEST QUALITY PRACTICABLE
FROM COPY FURNISHED TO DDC

THIS PAGE IS BEST QUALITY PRACTICABLE
FROM COPY FURNISHED TO DDC

0 card no. 18 contains nodes:
0 49 98 99 100
0

ext. conn. = 43
int. conn. = 14
st = 29

0 card no. 19 contains nodes:
0 56 62 63 68
0

ext. conn. = 61
int. conn. = 24
st = 37

0 card no. 20 contains nodes:
0 58 59 64 65 70
0

ext. conn. = 75
int. conn. = 37
st = 38

0 card no. 21 contains nodes:
0 61 91 95
0

ext. conn. = 36
int. conn. = 6
st = 30

0 card no. 22 contains nodes:
0 67 71 72 74 75
0

ext. conn. = 53
int. conn. = 29
st = 24

0 card no. 23 contains nodes:
0 69 77 88 92
0

ext. conn. = 69
int. conn. = 17
st = 52

0 card no. 24 contains nodes:
0 76 85 89 90
0

ext. conn. = 81
int. conn. = 14
st = 67

0 card no. 25 contains nodes:
0 79 80 83 94
0

ext. conn. = 78
int. conn. = 14
st = 64

0 * * * * * e n d o f c o m p l e t e d * * * * *

expected total number of time=30 08728

cost of mod. 1=	444,000
cost of mod. 2=	528,000
cost of mod. 3=	426,000
cost of mod. 4=	328,000
cost of mod. 5=	406,000
cost of mod. 6=	450,000
cost of mod. 7=	602,000
cost of mod. 8=	340,000

THIS PAGE IS BEST QUALITY PRACTICABLE
FROM COPY FURNISHED TO DDC

cost of mod. 6= 542,000
cost of mod. 10= 640,000
cost of mod. 11= 370,000
cost of mod. 12= 358,000
cost of mod. 13= 342,000
cost of mod. 14= 604,000
cost of mod. 15= 468,000
cost of mod. 16= 540,000
cost of mod. 17= 462,000
cost of mod. 18= 422,000
cost of mod. 19= 414,000
cost of mod. 20= 624,000
cost of mod. 21= 342,000
cost of mod. 22= 478,000
cost of mod. 23= 526,000
cost of mod. 24= 582,000
cost of mod. 25= 544,000

total cost of the modules= 11762

acquisition cost= 352860

no. of equipments to be procured= 30

expected no of failures in 10 years

mod. 1= 34.033

mod. 2= 36.687

mod. 3= 26.806

mod. 4= 20.446

mod. 5= 38.159

mod. 6= 22.469

mod. 7= 25.042

mod. 8= 23.100

mod. 9= 29.013

mod. 10= 49.590

mod. 11= 24.729

mod. 12= 18.948

mod. 13= 14.270

mod. 14= 34.348

mod. 15= 9.829

mod. 16= 35.005

mod. 17= 30.774

mod. 18= 26.227

mod. 19= 25.386

mod. 20= 36.976

mod. 21= 28.382

mod. 22= 32.088

mod. 23= 40.892

mod. 24= 30.958

mod. 25= 23.705

spares mix for this design 48 51 39 34 53 34 37 39 42 66
38 32 27 48 22 49 44 38 37 51
41 45 56 44 35

system av.= 0.85195

total cost for the spares requirement= 15211860

support cost= 15217435

the cost of this design is= 15570295

0 starting node: 3

0 card no. 1 contains nodes:

0 1 2 3 4

0

ext. conn. = 21

int. conn. = 27

st = -6

0 card no. 2 contains nodes:

0 5 42 96

0

ext. conn. = 25

int. conn. = 0

st = 25

0 card no. 3 contains nodes:
0 6 8 9 13
0

ext. conn. = 33
int. conn. = 25
st = 8

0 card no. 4 contains nodes:
0 7 12 17 18 21
0

ext. conn. = 54
int. conn. = 35
st = 19

0 card no. 5 contains nodes:
0 10 11 15 16 20
0

ext. conn. = 35
int. conn. = 38
st = -3

0 card no. 6 contains nodes:
0 14 19 23
0

ext. conn. = 24
int. conn. = 8
st = 16

0 card no. 7 contains nodes:
0 22 81 86
0

ext. conn. = 33
int. conn. = 3
st = 30

0 card no. 8 contains nodes:
0 24 29 33
0

ext. conn. = 35
int. conn. = 9
st = 26

0 card no. 9 contains nodes:
0 25 39 44 45 49
0

ext. conn. = 60
int. conn. = 22
st = 38

0 card no. 10 contains nodes:
0 26 27 28 31 32
0

ext. conn. = 52
int. conn. = 46
st = 6

0 card no. 11 contains nodes:
0 30 34 78
0

ext. conn. = 58
int. conn. = 9
st = 39

0 card no. 12 contains nodes:
0 35 66 73
0

ext. conn. = 31
int. conn. = 4
st = 27

THIS PAGE IS BEST QUALITY PRACTICABLE
FROM COPY FURNISHED TO DDC

0 card no. 13 contains nodes:
0 36 38 51
0

ext. conn. = 37
int. conn. = 15
st = 22

0 card no. 14 contains nodes:
0 37 41
0

ext. conn. = 60
int. conn. = 7
st = 53

0 card no. 15 contains nodes:
0 38 48 54 55 60
0

ext. conn. = 47
int. conn. = 37
st = 10

0 card no. 16 contains nodes:
0 40 84 97
0

ext. conn. = 51
int. conn. = 0
st = 51

0 card no. 17 contains nodes:
0 46 50 52 53 57
0

ext. conn. = 80
int. conn. = 43
st = 37

0 card no. 18 contains nodes:
0 47 98 99 100
0

ext. conn. = 51
int. conn. = 14
st = 37

0 card no. 19 contains nodes:
0 56 62 63 68
0

ext. conn. = 61
int. conn. = 24
st = 37

0 card no. 20 contains nodes:
0 58 59 64 65 70
0

ext. conn. = 75
int. conn. = 37
st = 38

0 card no. 21 contains nodes:
0 61 91 95
0
0

ext. conn. = 36
int. conn. = 6
st = 30

THIS PAGE IS BEST QUALITY PRACTICABLE
FROM COPY FURNISHED TO DDG

THIS PAGE IS BEST QUALITY PRACTICABLE
FROM COPY FURNISHED TO DDC

0 card no. 22 contains nodes:
0 67 71 72 74 75
0

ext. conn. = 53
int. conn. = 29
st = 24

0 card no. 23 contains nodes:
0 69 77 82
0

ext. conn. = 53
int. conn. = 10
st = 43

0 card no. 24 contains nodes:
0 76 85 89 90
0

ext. conn. = 81
int. conn. = 14
st = 67

0 card no. 25 contains nodes:
0 79 83 88 92
0

ext. conn. = 34
int. conn. = 28
st = 6

0 card no. 26 contains nodes:
0 80 87 93 94
0

ext. conn. = 94
int. conn. = 0
st = 94

0 * * * * * r u n c o m p l e t e d * * * * *
expected total maintenance time=41.65934
design not feasible

QUIT

r 1405 16.939 3.424 164 level 4, 33

logout

Biesel 9567c0016 logged out 11/25/77 1405.9 est Fri
CPU usage 33 sec, memory usage 61.1 units,
hangup