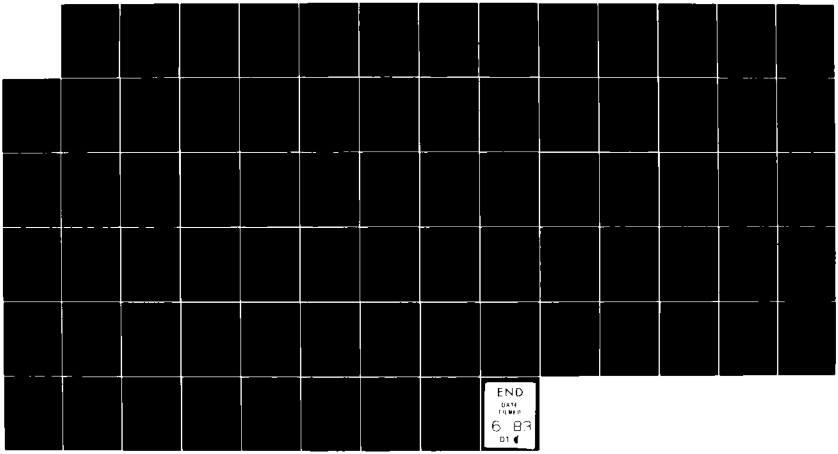
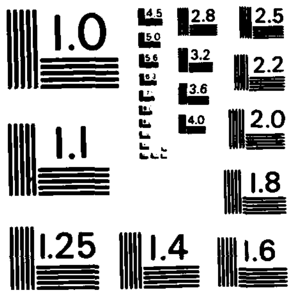
AD-A127 851 DISTRIBUTION OF FUNCTIONALITY AND LEVELS OF FUNCTIONALITY AS A SOLUTION T..(U) NORTH CAROLINA AGRICULTURAL AND TECHNICAL STATE UNIV GREENSBO.. 1/1

UNCLASSIFIED H L MARTIN ET AL. APR 83 ARO-18591.1-EL-H F/G 9/2 NL





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

Unclassified
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER 18591.1-EL-H	2. GOVT ACCESSION NO. AD A129 65	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) Distribution of Functionality and Levels of Functionality as a Solution to the CONOPS Problem		5. TYPE OF REPORT & PERIOD COVERED Final: 15 Aug 81 - 15 Feb 83
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Harold L. Martin Pakize S. Pulat		8. CONTRACT OR GRANT NUMBER(s) DAAG29 81 G 0009
9. PERFORMING ORGANIZATION NAME AND ADDRESS North Carolina Agricultural & Technical State U Greensboro, NC 27411		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS U. S. Army Research Office Post Office Box 12211 Research Triangle Park, NC 27709		12. REPORT DATE Apr 83
		13. NUMBER OF PAGES 73
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.

A

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

18. SUPPLEMENTARY NOTES

The view, opinions, and/or findings contained in this report are those of the author(s) and should not be construed as an official Department of the Army position, policy, or decision, unless so designated by other documentation

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

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

This document is the final report for a research project sponsored by the United States Army to study the distribution of functionality within a distributed system so as to provide the system with graceful degradation. In addition, overall system capabilities were studied in an effort to describe system performance as system resources were alleviated due to failure.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

83 05 03 023

AD A127651

Period: August 14, 1981 to February 15, 1983

Dr. Harold L Martin

Department of Electrical Engineering

Distribution Of Functionality And Levels Of Functionality As A Solution To The CONOPS Problem

Research Agreement No. DAAG29-81-G-0009

Period: August 14, 1981 to February 15, 1983

Dr. Harold L. Martin

Department of Electrical Engineering

And

Dr. Pakize S. Pulat

Department of Industrial Engineering

North Carolina Agricultural And Technical State University

Greensboro, North Carolina 27411

CONTENTS

		PAGE
1.0	Introduction	3
2.0	Research Description	4
	2.1 The Problem	5
	2.2 Brief Review of Literature	8
3.0	The Assignment of Modules to Processors	10
	3.1 Development of Model	10
	3.1.1 Assignment Model	14
	3.2 Extensions to Model	18
	3.3 Assignment of Additional Modules	19
	3.4 Scheduling Modules to Processors Using Network Flow Approach	22
	3.4.1 Flow-Chart for the Algorithm	24
	3.4.2 Example 1	25
	3.4.3 Example 2	29
4.0	Scheduling Algorithm	38
	4.1 Multiprocessor Scheduling Algorithm	38
5.0	Summary	42
6.0	References	44
7.0	Appendices	46
	Appendix I - Solution of the Assignment Problem	47
	Appendix II - Assignment of Extra Modules to Processors	52
	Appendix III - Network Flow Approach to two Processor Scheduling Example Problem	55
	Appendix IV - Network Flow Approach to Three Processor Problem	59
	Appendix IV-A 'JAN'; Processor S_1 is Active and the dummy processor S_4 is connected to processor S_2 & S_3.	
	Appendix IV-B 'JANA'; Processor S_2 is active and the dummy processor S_4 is connected to processor S_1 & S_3.	
	Appendix IV-C 'JANAK'; Processor S_3 is active and the dummy processor S_4 is connected to processor S_1 & S_2.	

LIST OF FIGURES

		PAGE
Figure 2.1	The proposed model of the distributed system.	7
Figure 3.1	The distributed processing system illustrating the assigning mechanism.	11
Figure 3.2	The example of the ideal balanced assignment strategy.	13
Figure 3.3	An example to the network representation of the problem.	23
Figure 3.4	Example of a multicut.	23
Figure 3.5	Intermodule communication graph.	26
Figure 3.6	Intermodule and module-to-processor communication graph.	27
Figure 3.7	The solution to the two processor problem.	28
Figure 3.8	The module interconnection network.	30
Figure 3.9	Illustration of processor S_1 active and the dummy processor S_4 connected to S_2 and S_3.	33
Figure 3.10	Illustration of processor S_2 active and the dummy processor S_4 connected to S_1 and S_3.	34
Figure 3.11	Illustration of processor S_3 active and the dummy processor S_4 connected to S_1 and S_2.	35
Figure 3.12	Optimal module to processor schedule.	37

ABSTRACT

This document is the final report for a research project sponsored by the United States Army to study the distribution of functionality within a distributed system so as to provide the system with graceful degradation. In addition, overall system capabilities were studied in an effort to describe system performance as system resources were alleviated due to failure.

1.0 INTRODUCTION

The currently existing systems employed by the military, which consist of a function or functions residing on a single processing unit, fail in the worst possible way in the field. That is, when the processing unit is destroyed, the function ceases to exist. This, coupled with the impossiblilty of reverting to a manual method - since much of the doctrine for the function becomes embedded in the automated system - leads to an intolerable situation in the field. In essence, the operational capability of a unit can be destroyed by destroying its computer.

A method of supplying systems which differ in a fundamental way from the current "point system" approach was studied. This method is based upon a combination of distribution of the functionality of a system across a network of processing units and provisions of each function (and the subfunctions of which it is made of) in a number of locations in the network with a number of levels of functionality. In operation, the request for a function causes the best instance of that function currently available to the requesting processor to be invoked. Graceful degradation may be provided by making this best function dynamic in the sense that, depending upon the currently existing system configuration, the function can be rescheduled if it was lost during the previous scheduling attempt or if the scheduled hardware is currently unavailable. Clearly, such an approach would ensure the availability of computer capability over a more extended period of time than the single stand-alone computers presently being employed. Further, there are other advantages to utilizing such a distributed system. First, a more powerful capability can be made available by interconnecting several computer systems together in a distributed fashion. Secondly, the computer systems used need not be all that expensive since the distributed system will require that functions be segmented in such a way that they can run on a less powerful and less expensive machine. Thirdly, since the system modules will be assumed to be identical for this research (but in reality may or may not be), it is much easier and more economical to replace any faulty or destroyed modules.

2.0 RESEARCH DESCRIPTION

In recent years distributed processing systems have been a subject of interest due to the availability of computer networks and the availability of microprocessors for use in inexpensive distributed computers. Therefore, this study attempts to develop a scheduling algorithm for a distributed system, to improve the total system performance and to effectively utilize all system resources. The objective of the scheduling algorithms is to minimize the processing time of the system, balance the load among the available processors and to increase the efficiency of the total system. This study also attempts to develop a rescheduling algorithm which provides a means of regaining a function when it has been lost due to loss of a particular constraint attached to the distributed systems. The introduction of the microprocessor has made distributed processing an increasingly popular notion in the computer industry. Economics of fabrication have substantially reduced the cost of replacing processors in a system, making distributed processor systems economically attractive. Microprocessors have created an environment that is fostering the growth of distributed computation.

By a distributed computer system in which there exist several progammable processors, and inwhich typical computations visit two or more processors during an execution. The distributed programs that we consider in our analyses are assumed to be made up of modules that are, in general faceed to reside on any processor if it is not an attached module, in the distributed system.

Distributed computer systems appear to offer extensibility improvements over these configurations due to decentralization of the interconnection and control logic (both hardware and software). As the system is scaled up in size, nonlinearities and boundary conditions in performance are less likely than for centralized systems.

2.1 The Problem

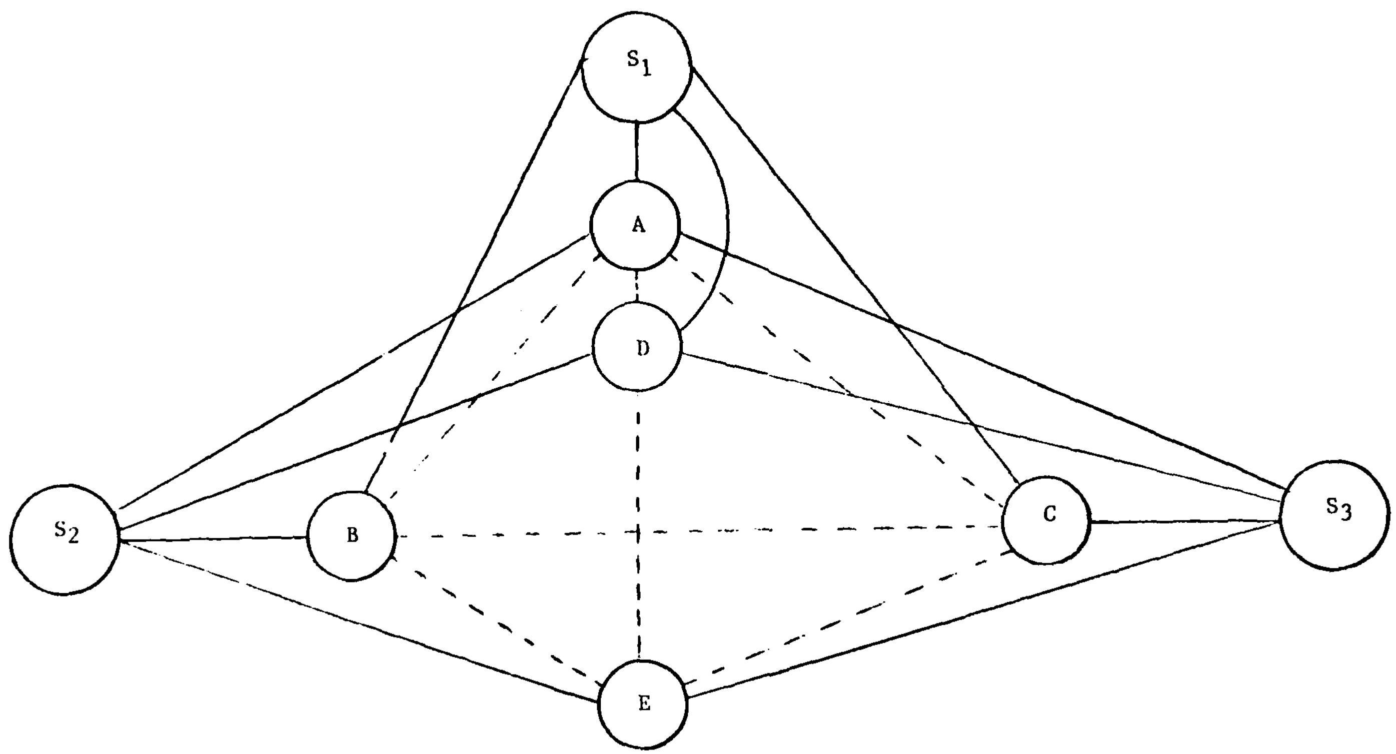
The existing systems employed by the military, consist of a function or functions residing on a single processing unit with the result that when the processing unit is destroyed, the function ceases to exist. This coupled with the impossibility of reverting to a manual method-since much of the doctrice for the function becomes embedded in the automated system-leads to an intolerable situation in the field.

A method of supplying systems which differ in a fundamental way from the current "point systems" approach is proposed for investigation.

This method is based upon a combination of distribution of the functionality of a system across a network of processing units and provisions of each function (and the sub-functions of which it is made of) in a number of locations in the network with a number of levels of functionality. In operation, the request for a function causes, the best instance of that function currently available to the requesting processor to be invoked. Depending upon the currently existing system configuration, the function can be rescheduled if it was lost during the previous scheduling attempt or if the schedule hardware is currently unavailable. Clearly, such an approach would ensure the availability of computer capability over a more extended period of time than the single stand-alone computers presently being employed.

The proposed model of the distributed system to be considered in this research is shown in Fig. 2.1. Although the system is depicted with only five modules and 3 processors there may be many more, or even less modules. Each module is defined to have a set of resources, such as CPU speed, memory size, computational power, etc. These resources may or may not be the same for each module. The solid arrows indicates the communications link over which programs and data are transferred between the various modules. The dashed arrows indicate that a communications link could possibly exist between the modules as shown.

As stated earlier, the number of modules and the exact interconnection of these modules may vary depending upon the configuration that supports the most efficient scheduling algorithm. Therefore, in the project we investigated and developed a heuristic that will schedule functions to modules subject to constraints that are well defined.



—— Communication links between modules and processors.

- - - Possible communication links between modules.

Figure 2.1 The proposed model of the distributed system.
S_1, S_2, and S_3 - Processors, A, B, C, D, and E - Modules

2.2 Brief Review of Literature

Since the mid-1960's several investigations have been done in the area of distributed computer systems. Distributed systems has received increased attention in the recent literature. Some of this work will be discussed briefly next.

Harold S. Stone [5] has shown how the program, modules of a program may be assigned to the processors in a distributed computer system so as to minimize the overall cost, including two types of cost: the cost of running an individual's module on a processor and the cost of interprocessor communication that arises in the event of transfer of control of execution from one processor to another.

V. B. Gylys and J. A. Edwards [4] introduced a performance measure for a real time-distributed network and discussed computational techniques for obtaining optimal work load partitioning over a network configuration. Optimality is attained by the assignment of programs of computers which minimizes the intercomputer bus traffic, subject to constraints on the maximum loading of each computer. This principle could be used to determine workload partitioning both at design and in real-time; furthermore, at design time, it can also be applied to finding optimal network configuration for a given software design. Gylys and Edwards, proceeded with formulation of an optimally criterion for workload distribution and derived a mathematical optimization problem; subsequently it examined the computational techniques for solving that particular problem; it ended with a critical assessment of the proposed method.

The problem was to find an assignment of program modules to the processors in the network. We researched to make such an assignment efficient by using the method of Ford and Fulkerson [8] that has been developed for maximizing flows in commodity networks. The maximum flow algorithm was extended to solve the multi-processor models. This is to say that the value of a maximum flow in a commodity network is equal to the weight of a minimum weighted cutset of the network. A cutset of the commodity network is a set of edges which when removed disconnects the source nodes from the sink nodes. This is explained in detail in Chapter II.

W. W. Chu, L. Y. Holloway, M. J. Lan, and K. Efe, [10] concentrated on the problem of task allocation in distributed data processing. A distributed processing system has conflicting requirements and this paper therefore, made a compromise to find the optimum assignment policy for a task. Different approaches for solving the assignment problems have been surveyed. All of the possible methods for partitioning a task have not yet been fully investigated, although some promising attempts have been reported.

V. Balanchandra, J. W. McGredie, and O. I. Mikhai [1] investigated the job assignment problems in a network of non-identical but functionally similiar computers. Periodic review models are formulated utilizing (0-1) integer programming, network flow algorithms, transportation problems and heuristic balancing procedure. They investigated the power of each; to determine what type of information about job requirements is needed; to compare the processing requirements and the quality of the solution for each formulation.

Edward K. Bowdon, Sr. [3] has done research, aimed at developing analytical tools for system modeling and analysis of real-time computer networks. He formulated an idealized mathematical model for multiserver systems with a finite length nonpreemptive priority queue. Given that jobs consist of dependent tasks having linear loss functions, Bowdon formulated an algorithm for assigning priorities to tasks. He defines a feasible successor set of tasks as a subset of tasks which can be scheduled independently. Each task is weighted by the maximum cost rate per task over every feasible successor set of the task and the task set is divided into levels based on the precedence relationship among the tasks. Generally, the algorithm gives priority to tasks within a given level. This algorithm is in general, suboptimal.

Kennal Efe [7] extended the work done by Chu, Holloway, Lan and himself to find a heuristic for task allocation in a distributed system. The purpose of his study for task allocation scheduling in a set of interconnected processors was to reduce job turnaround time. This was done by minimizing any communication between processors. A distributed processing system has conflicting requirements; therefore some compromises were made in order to find the optimal assignment policy for a task.

3.0 THE ASSIGNMENT OF MODULES TO PROCESSORS

3.1 Development of Model

Distributed processing enhances system(s) performance by employing several processors to handle the processing load. A representation of the distributed processing system is shown in Fig. 3.1. The key elements in

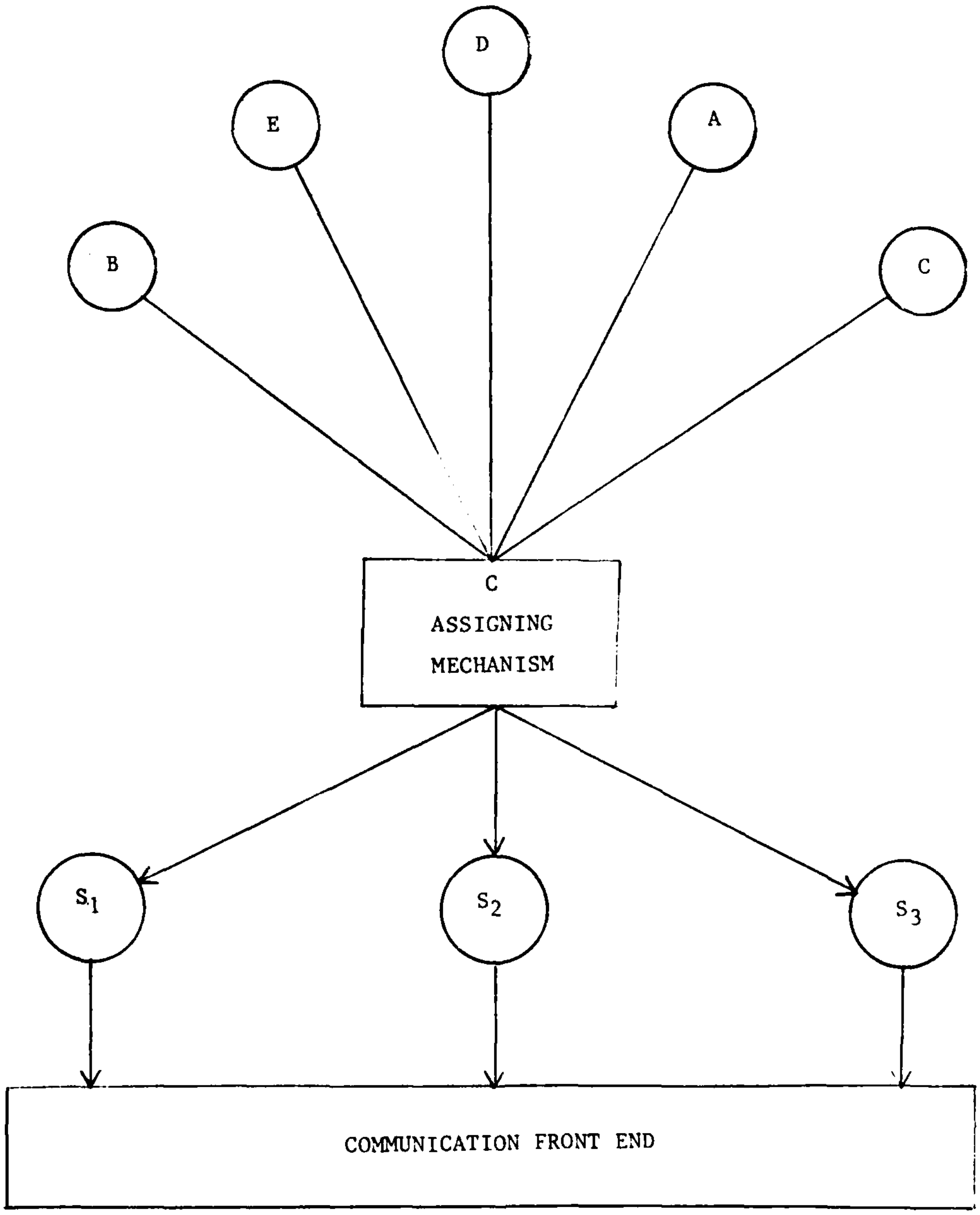


Figure 3.1 The distributed processing system illustrating the assigning mechanism.

this system are a set of modules to be processed $\{i_1, \ldots i_2 \ldots i_m\}$ and a module allocation or a module assignment mechanism, C, which assigns each of the i modules to one of the n processors, $\{S_1, S_2 \ldots S_n\}$. In general, the number of modules is much higher than the number of processors. There can be a mechanism where the processors in this environment communicate among themselves via interconnection mechanism shown in Fig. 3.1. Modules may be assigned to different processors for the fastest processing time.

The intermodule communication between any pair of modules is determined by software design and fixed attribute of the modules at the time of module assignment. We have to assign modules to processors so that all processors are approximately evenly loaded. The example of the ideal balanced assignment strategy is exhibited in Fig. 3.2 where six modules have to be processed by three processors, i.e., these are the modules $\{i_1, i_2, \ldots i_6\}$ which are to be assigned among the three processors $\{S_1, S_2, S_3\}$. We assume each module has identical processing requirements and processing time, and that each processor has identical processing abilities. For simplicity, let us also assume that the processing time of each processor is one minute per module, i.e., for the case illustrated in Fig. 3.2, the system is able to process the module assignment in two minutes.

Figure 3.2 exhibits the following:

S_1 will process i_1 and i_4

S_2 will process i_2 and i_5

S_3 will process i_3 and i_6

Thus the required total time equals two minutes.

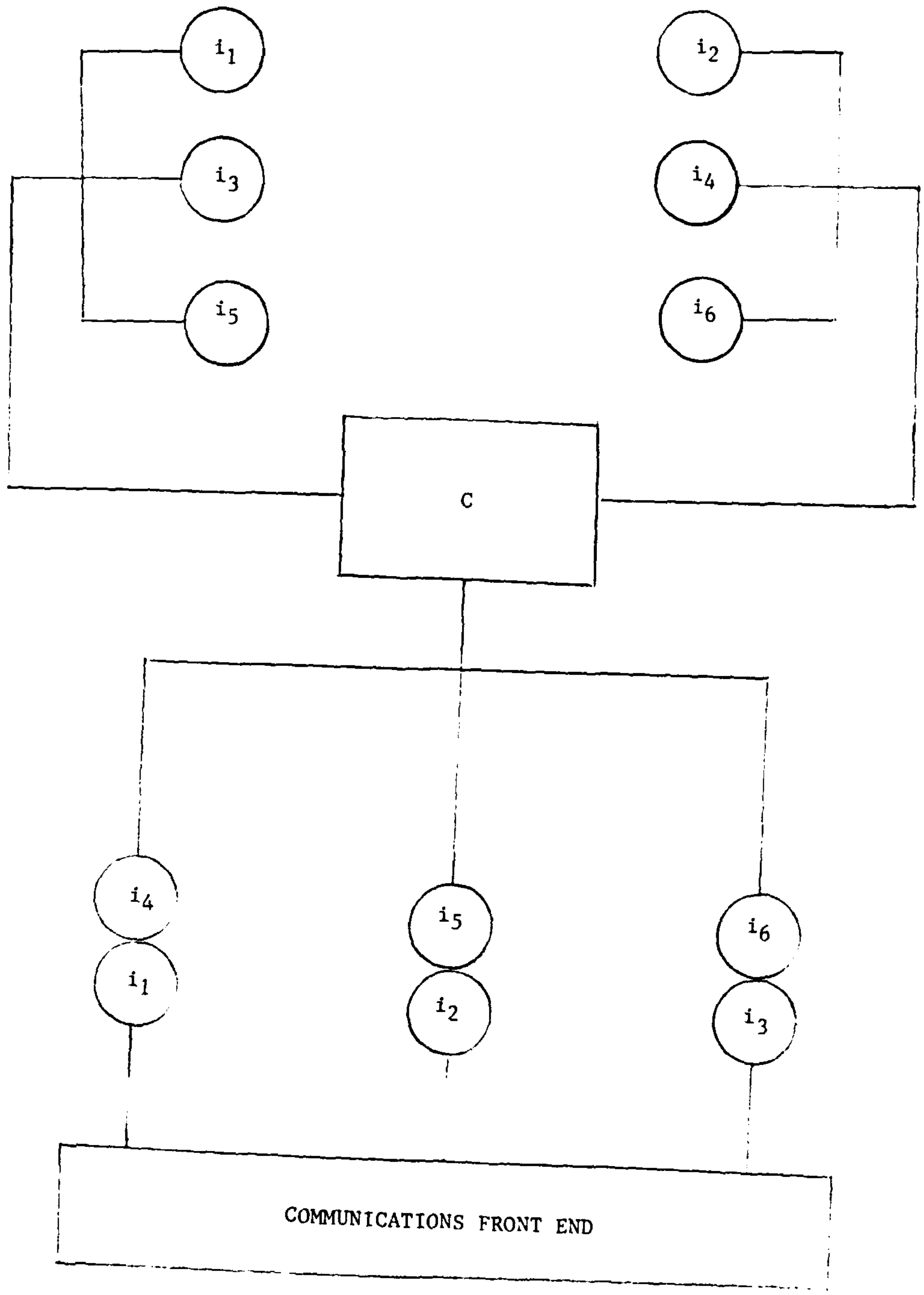


Figure 3.2 The example of the ideal balanced assignment strategy.

The objective is to distribute modules in a manner such that maximum number of modules can be processed simultaneously for the maximum system performance.

The method used is a graphical method where modules to be assigned are like a set of nodes in a network. We assume the inter-module communication time between the modules are known, represented by the time unit or by the weight of undirected arcs connecting the nodes. An inter-module or module to processor communication of zero means that no communication takes place between the two modules or between the module to processor. They are connected in the network to show that there could be communication. An inter-module or module to processor communication time of infinity means these modules should not be processed by that particular processor. We also assume that all processors are ready and available at all times for an assignment.

Therefore, the assignment strategy in this model is to minimize total processing time defined as the sum of the processing time with respect to the module connection to the processor. In order to represent the inter-module communication and/or module to processor assignment, we propose the following assignment model.

3.1.1 Assignment Model

Given the network configuration the objective is to assign modules to processors with the objective of minimizing total time where total time is the running time of modules in the processors plus the intercommunication time among the modules. The problem can be formulated as a linear programming problem as following:

Define X_{ij} = 1 if module i is assigned to processor j, 0 otherwise.

Let t_{ij} = running time of module i in processor j

$t_{i_k i_\ell}$ = intercommunication time for modules i_k and i_ℓ

I = set of modules, {1, 2, ...,m}.

J = set of processors, {1, 2, ...,n}.

The model can be stated as:

$$\text{Minimize} \quad \sum_{i \varepsilon I} \sum_{j \varepsilon J} t_{ij} X_{ij} + \tfrac{1}{2} \sum_{j \varepsilon J} \sum_{i_k i_\ell \varepsilon I} t_{i_k i_\ell} \omega_{i_k i_\ell j}$$

$$(1) \qquad + \tfrac{1}{2} \sum_{j \varepsilon J} \sum_{i_k i_\ell \varepsilon I} t_{i_k i_\ell} y_{i_k i_\ell j}$$

subject to

$$(2) \qquad \sum_{i=1}^{m} X_{ij} \geq 1 \qquad \text{for all } j \varepsilon J$$

$$(3) \qquad \sum_{j=1}^{n} X_{ij} = 1 \qquad \text{for all } i \varepsilon I$$

$$(4) \qquad X_{i_k j} - X_{i_\ell j} + \omega_{i_k i_\ell j} - y_{i_k i_\ell j} = 0 \quad \text{for all } (i_k, i_\ell) \text{ and } j$$

$$X_{ij} \geq 0 \qquad \text{for all } i \varepsilon I \text{ and } j \varepsilon J$$

The above module assigns at least one module to each processor with the objective of minimizing total running time. If there exists no constraint as to the utilization of processors (i.e., not all the processors need to be used), then one can simply remove the first constraint set from the above model before solving the problem.

To clarify the procedure, the following simple, five modules, three processors problem is modeled as an assignment problem and solved using the available LINDO (Ref. 9) package program. (See Appendix I).

Tables I and II give the intermodule communication times and module to processor communication times, respectively.

The optimal assignment as read from the computer output is as follows:

Module	Processor
A	1
B	2
C	3
D	3
E	3

Total running time = 33 time units.

TABLE I. INTER MODULE COMMUNICATION TIME

MODULE	A	B	C	D	E
A		2	3	3	-
B			2	2	1
C				2	7
D					
E					

TABLE II. MODULE TO PROCESSOR COMMUNICATION TIME

MODULE	S_1	S_2	S_3
A	3	12	15
B	10	4	12
C	12	14	4
D	10	7	5
E	4	7	3

3.2 Extensions to Model

The above model assigns modules to processors with the objective of minimizing total running time. It assumes that:

(a) Each processor must have at least one module assigned to it.

(b) Each processor has enough capacity to handle all the modules assigned to it.

(c) There exists only one of each type of module in the system.

(d) Workload of the processor is not an issue.

The above assumptions can be relaxed as the assignment model can be modified to incorporate the changes. To relax the first assumption, one simply removes (2) which is the first constraint set from the model. To relax the second assumption, one needs to introduce a new constraint set indicating that the memory space needed by the modules assigned to a specific processor must not exceed the total available memory space in the processor. Let m_i denote the memory space required by module i and M_j denote the total amount of available space in the jth processor. Then,

$$\sum_{i=1}^{n} m_i X_{ij} \leq M_j \qquad \text{for all } j \varepsilon J.$$

It should be noted that the inclusion of this constraint set to the existing model destroys the topology of the model. One can not now guarantee integer solutions. Therefore, one needs to use some other solution methods, like integer programming methods, to restrict X_{ij} variables to have values of 0 or 1, which decrease the efficiency of the model considerably. A similiar problem, called the job scheduling problem, with no communication between the jobs has been efficiently solved as the knopsach problem and presented at the ORSA/TIMS San Diego Conference (October 1982). The existence of intermodule communication in our model prevents us to use the above mentioned algorithm.

The third assumption is the subject of the proceeding section. The last assumption can be relaxed by adding another set of constraints into the model which turn decreases the efficiency of the proposed solution. A more realistic approach would be to get the assignments neglecting the workloads and then reschedule modules to processors in a way to balance the workload with the objective of minimizing the increase in the total processing time.

3.3 Assignment of Additional Modules

The utilization rate of modules plays an important role in the scheduling process. Scheduling highly used modules to one processor increase the queue length for that procesor and in turn decreases the realiability of the whole system. Duplicates of the module must be assigned to other processors to resolve this issue. How many of each module and to which processors to be assigned are the concern of this section.

Let P_i denote the utilization rate of module i, which is a predetermined number. The expected number of module i, E(i), can be calculated by multiplying the utilization rate of module i by the available number of processors. That is,

$$E(i) = P_i \times n$$

The procedure that we propose is the following: neglecting the extra modules. Solve the assignment problem as explained in section 2.1 and get the initial assignments. Then, calculate E(i) for each model. The assignment of extra modules to the processors will be obtained by solving the following linear programming model.

$$\text{Minimize} \quad \sum_{i \in I} \sum_{j \in J} t_{ij} X_{ij}$$

$$\text{Subject to} \quad \sum_{j} X_{ij} = E(i) - 1 \qquad \text{for all } i$$

$$X_{ij} = 0 \quad \text{if module } i \text{ is assigned to processor } j \text{ in the previous assignment problem}$$

$$X_{ij} \leq 1 \quad \text{for all } i \text{ and } j.$$

where t_{ij} is the processing time of module i in processor.

The above program assigns the extra modules to processors with the objective of minimizing total processing time.

Table III gives the utilization probability for each module and the expected number of modules required from each module for the problem given in Table IV and E_i is taken to be the smallest integer greater than or equal to the product P_i x n.

TABLE III

MODULE	P_i	E_i
A	.2	1
B	.5	2
C	.8	3
D	.4	2
E	.7	3

The assignment model and the solution for the above problem is given in Appendix II. Table IV summarizes the solution.

TABLE IV

MODULE	# OF MODULES	PROCESSOR (S) ASSIGNED	TIME REQUIRED
A	1	S_1	3
B	2	S_1, S_2	14
C	3	S_1, S_2, S_3	30
D	2	S_2, S_3	12
E	3	S_1, S_2, S_3	14

3.4 Scheduling Modules To Processors Using Network Flow Approach

There exists a close relationship between assignment and network flow models. Simply, one is the dual of the other. Therefore, the scheduling problem can be attacked as a network flow problem in the following way.

Let the processors and modules be the nodes of the network. A set of arcs connect modules to modules and modules to processors. If arc (i,j) connect module i to module j then t_{ij} indicate the intercommunication time between modules i and j. On the other hand, if arc (i,j) connects module i to processor j then t_{ij} is calculated as

$$t_{ij} = \sum_{\substack{k \neq j \\ k \varepsilon J}} \frac{t_{ik} - t_{ij}}{n-1}$$

that is, t_{ij} is the sum of processing times of module i in the processors other than j less the processing time of module i in processor j divided by number of processors minus one. Figure 3.3 is an example to such network configurations. The problem of assigning modules to processors with the objective of minimizing total time can now be translated to the problem of finding a multicut with minimum value. The set of modules in the subset of a processor will be assigned to that processor.

The procedure for n processor m module problem can be summarized as follows:

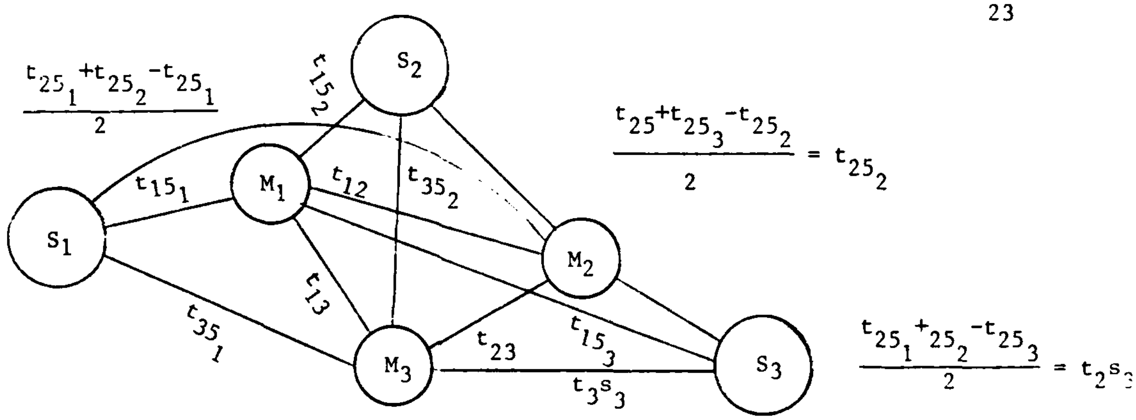


Figure 3.3 An example to the network representation of the problem.

Definition: A multicut partitions the graph into n disjoint sets where each set contains one and only one processor. No proper subset of this cut is also a multicut.

Figure 3.4 is an example of a multicut for the given network.

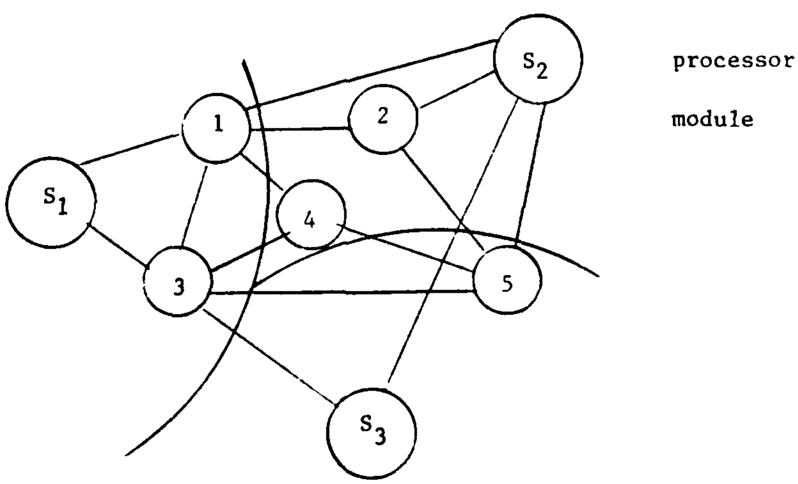


Figure 3.4 Example of a multicut.

1. Construct the general network.

2. For each processor S_i, find the maximum flow from processor S_i to all other processors. Store the minimal cut. At this step n maximum flow problems will be solved and n minimum cuts will be located.

3. Form n-1 multicuts.

4. The multicut with the minimal value generates the desired assignment.

3.4.1 Flowchart For The Algorithm

I. INITIALIZATION

Given the network configuration, read and store the data.

Set MIN = 0, KK = 0

II. MAXFLOW

For each processor S_i: Maximize the flow from S_i to all other processors (LINDO software package is used at this step). Retain the minimal cut, C_i.

III. MULTICUT

Using the n minimal cuts, (C's) generate (n-1) multicut. TT_i denotes the value of the minimal cut C_i, i.e., the time of running modules in the assigned processor (s). $TTMC_j$ denotes the value of the jth multicut.

IV. MINIMUM MULTICUT

Among the n generated multicuts locate the cut with minimum capacity.

3.4.2 Example 1. Two-Processor Problem

Consider the network of Fig. 3.5. The inter-module communication times for this network are shown in Table V. The module-to-processor communication times are shown in Table VI.

The objective is to minimize the total absolute running time of a program in the network. The network in Fig. 3.6 can then be constructed with nodes for each of the modules and inter-module communication times on the arc joining the nodes. In order to represent the processing time, two additional nodes are added to the network to represent the two available processors S_1 and S_2. The running time of each module on processor S_1 is denoted by the arc joining that module node to the node S_1. Similiarly, running time of each module on processor S_2 is denoted on the arc joining that module node to the S_2.

If a max-flow-min-cut algorithm [8] is performed on the network of Fig. 3.6, the cut shown by the heavy dark line on Fig. 3.7 is obtained. As shown in Fig. 3.7, modules A, B, C, D, and E are assigned to processor S_1 while module F is assigned to processor S_2 (See Appendix III for computations).

While this method is attractive in its simplicity, it has several limitations. The basic min-cut solution provides for a minimum time assignment between two processors. In general, an execution of this method to an arbitrary number of processors requires an N- dimensional mini cut algorithm, which quickly becomes computationally intractable. This limits the usefulness of the method in many applications. An

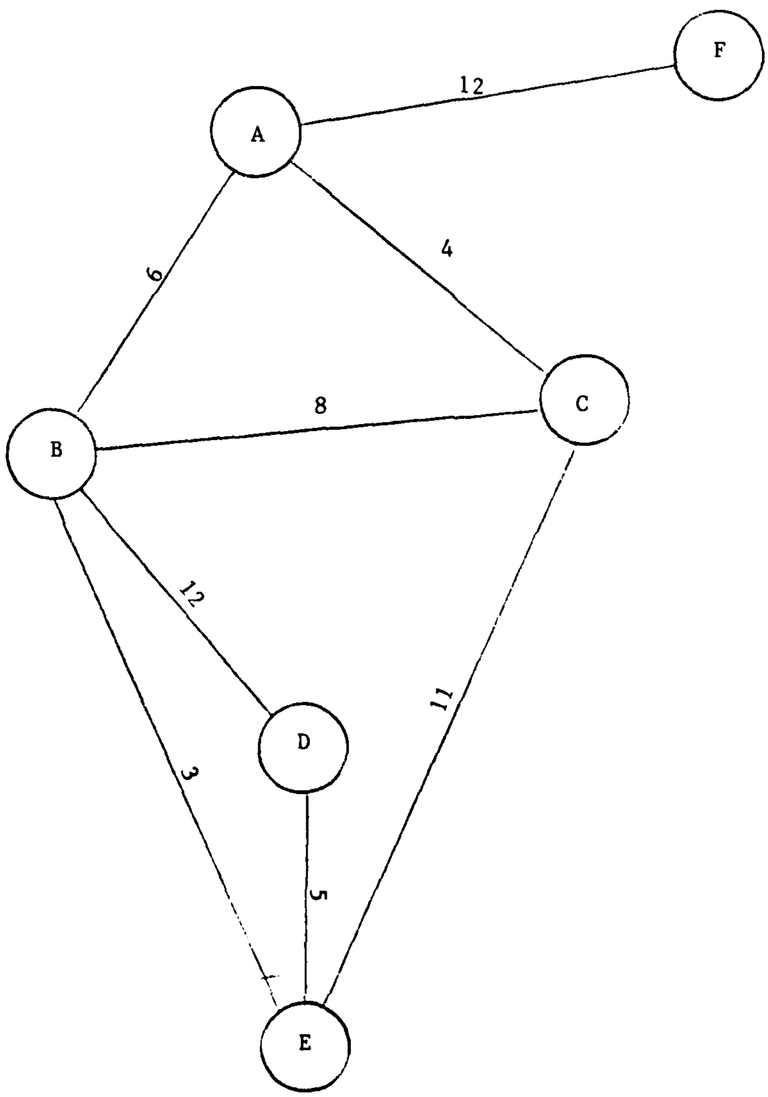


Figure 3.5 Inter-module communication graph.

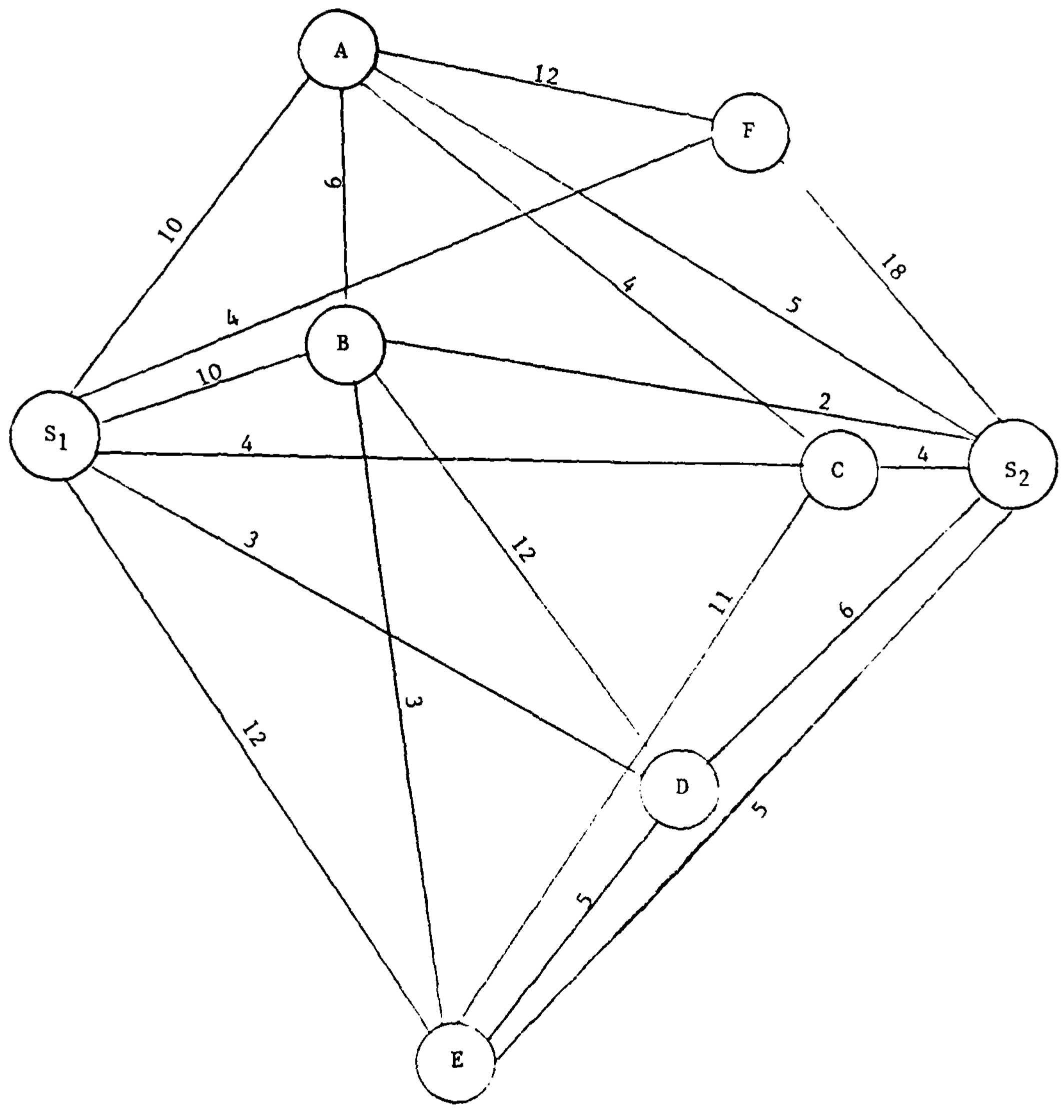


Figure 3.6 Intermodule and module-to-processor communication graph.

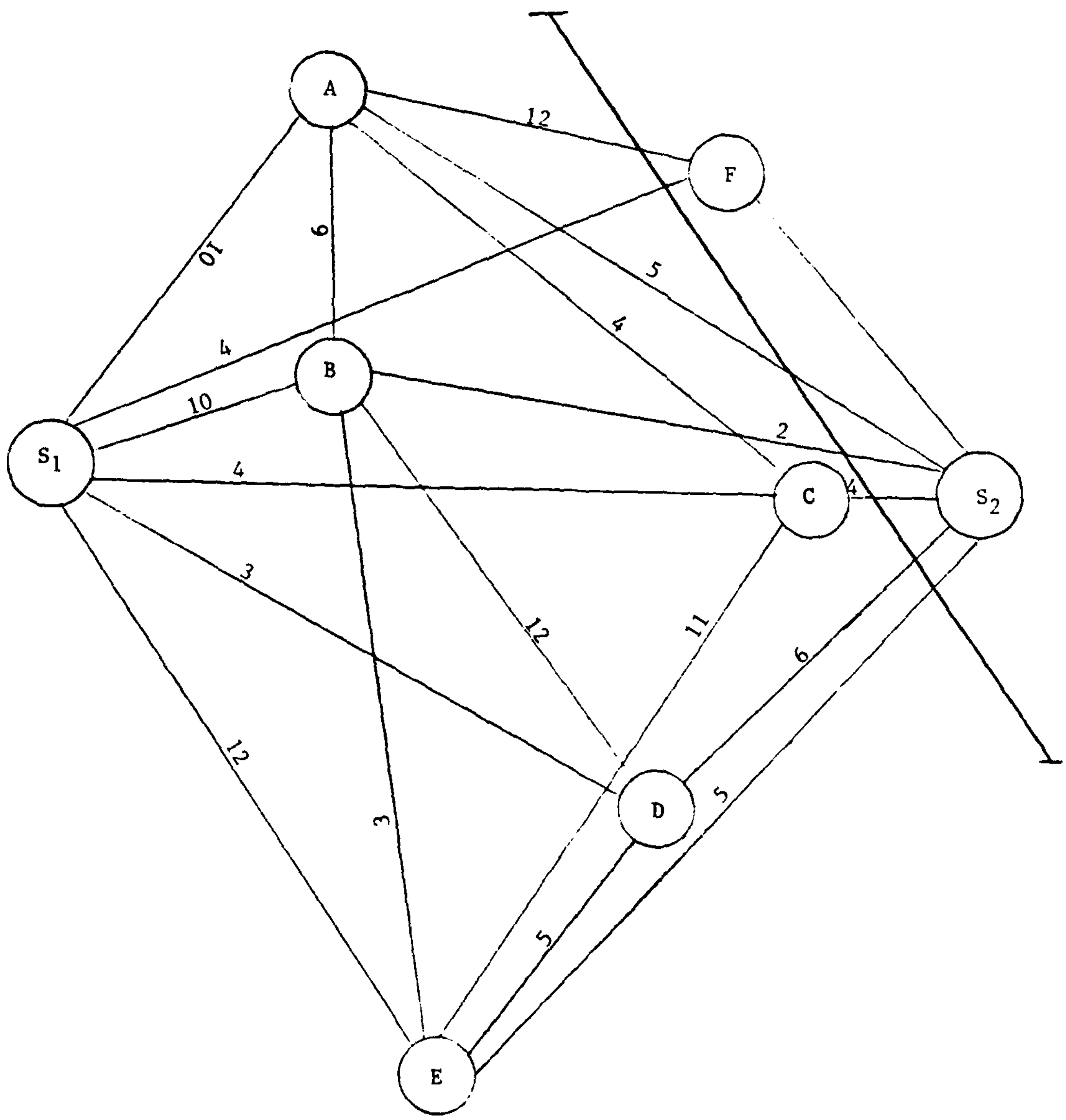


Figure 3.7 The solution to the two processor problem.

extension of this method to allow assignments among three processors or more is now proposed.

3.4.2 Example 2. Three-Processor Problem

The interconnection graph for a three-processor and five module assignment is shown in Fig. 3.8. This network is used with undirected arcs to find the relationship between processors and modules. Our research has shown that the maximum flow algorithm finds an optimal position for a three-processor system.

Table VII shows the time each module will take with a given processor. Therefore, when the program execution begins, the floating modules will be assigned to the processor which will process where the computation time is minimum. As an example, the best guess for module A will be processor S_1 and for module B will be processor S_2, etc. The weight of a branch is the total time charged to intermodule references represented by the branch.

Therefore, if k references between two modules occur during the running of a program and each reference takes t seconds, then when the modules are assigned to a particular processor, the weight of the branch representing these references is kt.

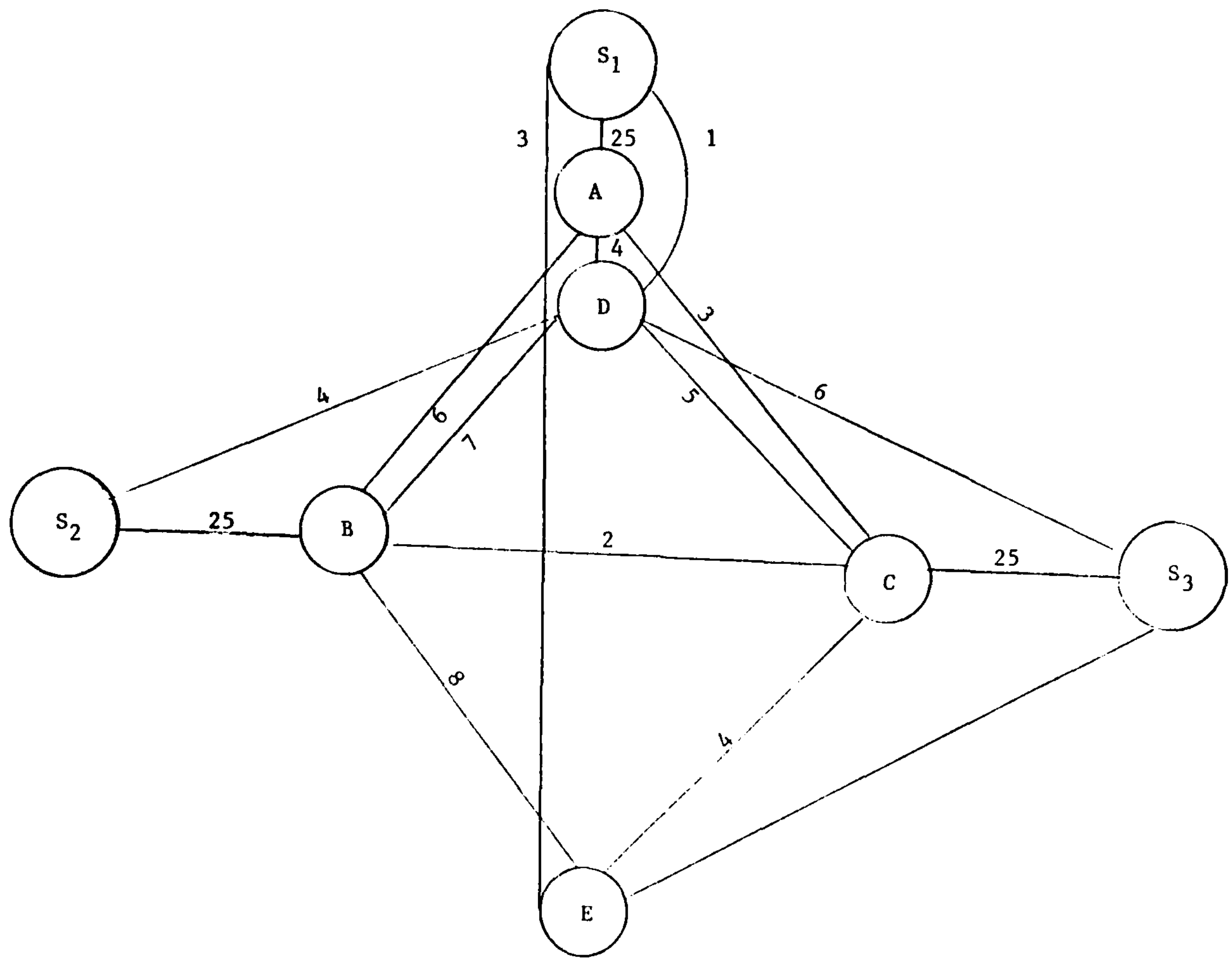


Figure 3.8 The module interconnection network.

TABLE VII. THE TOTAL RUNNING TIME OF THE MODULES ON A GIVEN PROCESSOR

MODULE	S_1	S_2	S_3
A	4	25	25
B	25	6	25
C	25	25	7
D	10	7	5
E	4	7	3

Since our objective is to minimize the total running time of a program, on a given processor, the only time factor taken into consideration is in Table VII, which gives the total running time on each processor. Also, in a distributed computing system, there is no parallelism of module execution within a program. Therefore, Table VII gives the total running time of the modules on their assigned processors. The problem was set having undirected arcs.

By solving the maximum flow problem on the network one gets the minimum weight cutset which determines the module assignment. This indicates that an optional assignment can be found by running a maximum flow algorithm on the network. The cutset will be defined later.

At this point, the dummy processor S_4 with infinity flow going into two of the processors is added. This means that when processor S_1 is active the infinity flow from S_4 will go to processor S_2 and processor S_3. Infinity flow indicates that the module cannot be assigned to that particular processor. This was done to each processor in the problem (See network in Figs. 3.9, 3.10, 3.11).

The flow was minimized via the constraints related with the modules and its intermodule references represented by the branches. When S_1 was selected to be the active processor and the dummy processor S_4 was connected to S_2 and S_3 respectively, the cut-set was 17. This includes the branches BA, ES_1, DA, CA and DS_1. (See Fig. 3.9 and computer print-out in Appendix IV-A).

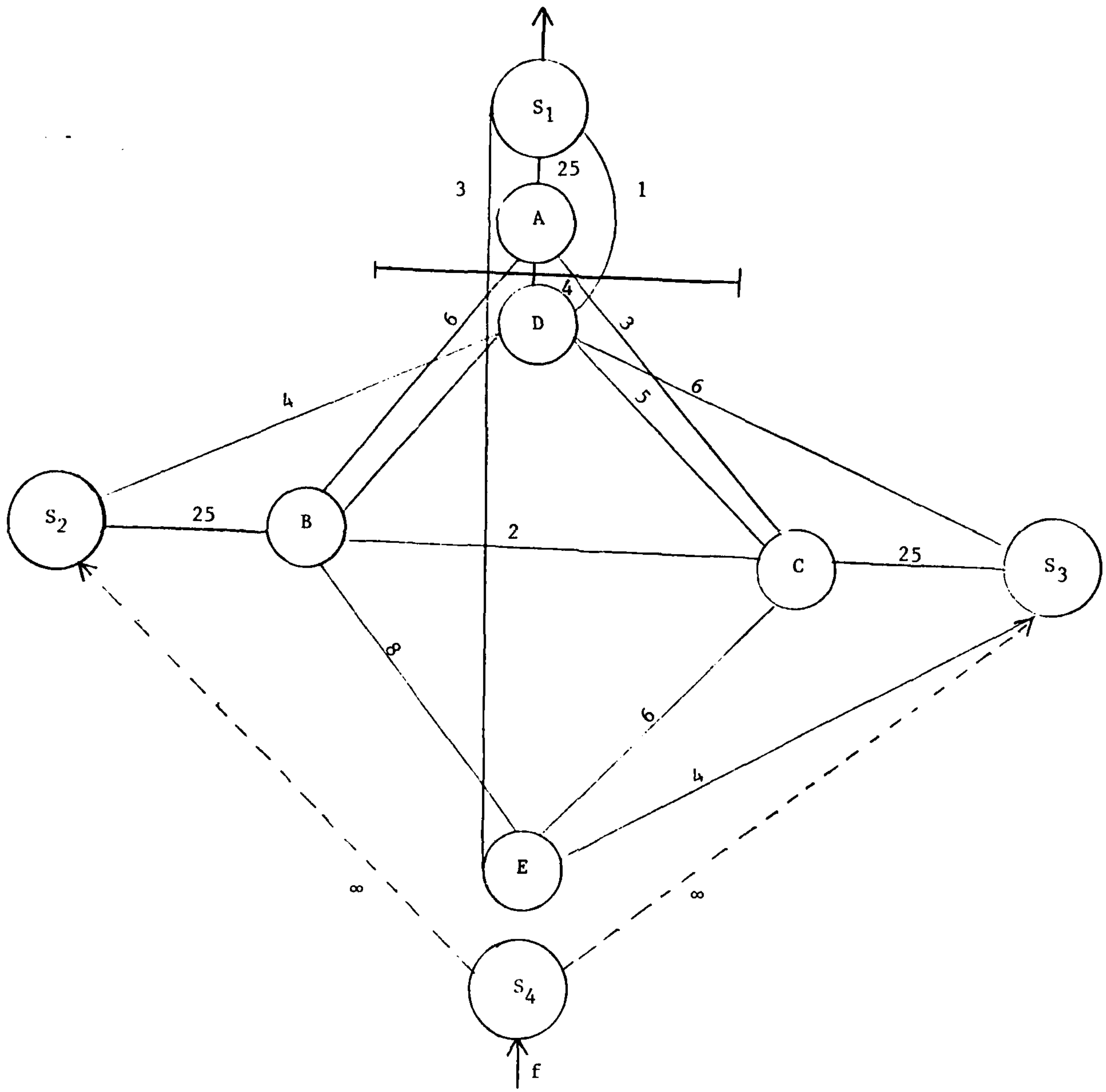


Figure 3.9 Illustration of processor S_1 *active and the dummy processor* S_4 *connected to* S_2 *and* S_3.

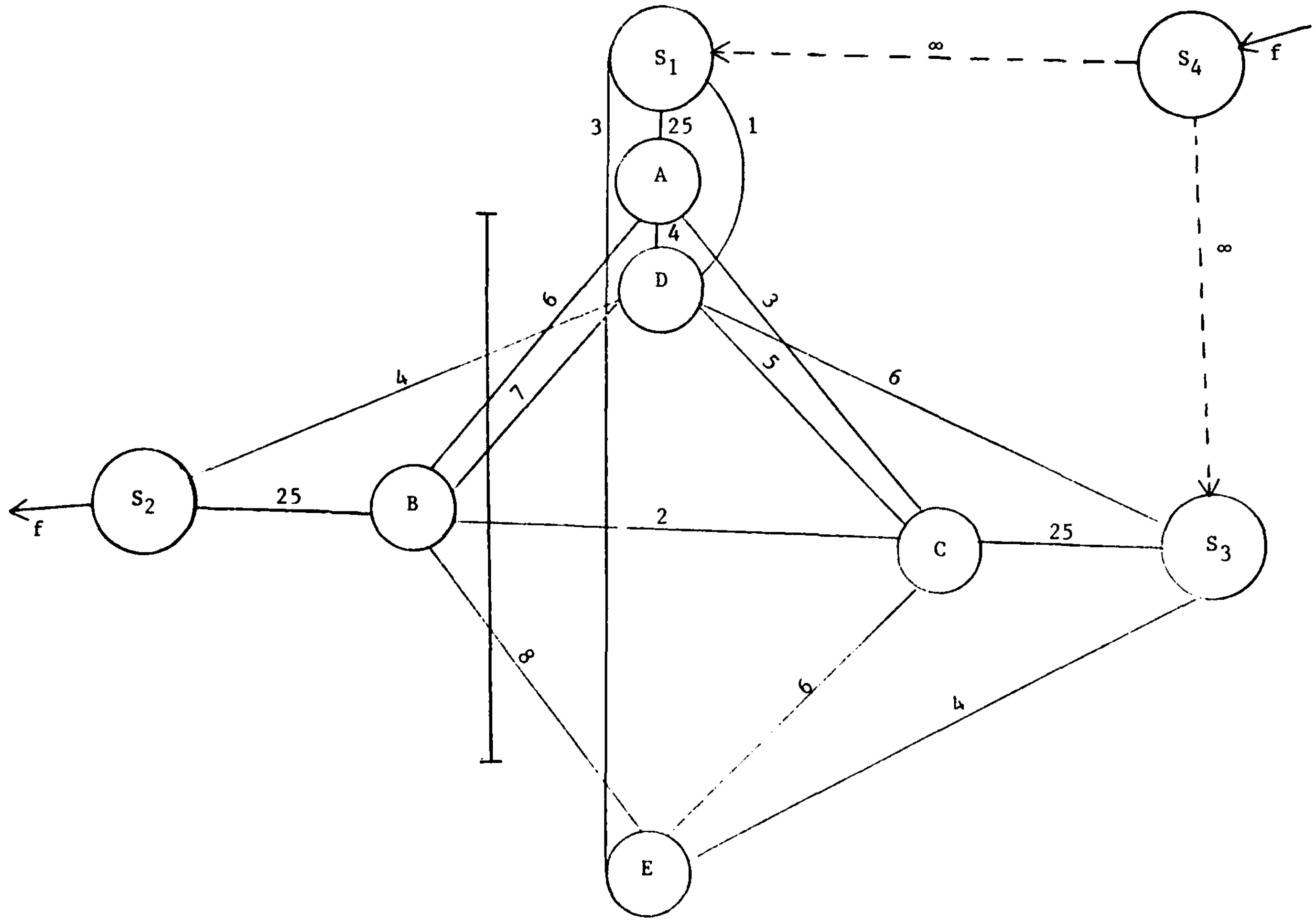


Figure 3.10 Illustration of processor S_2 active and the dummy processor S_4 connected to S_1 and S_3.

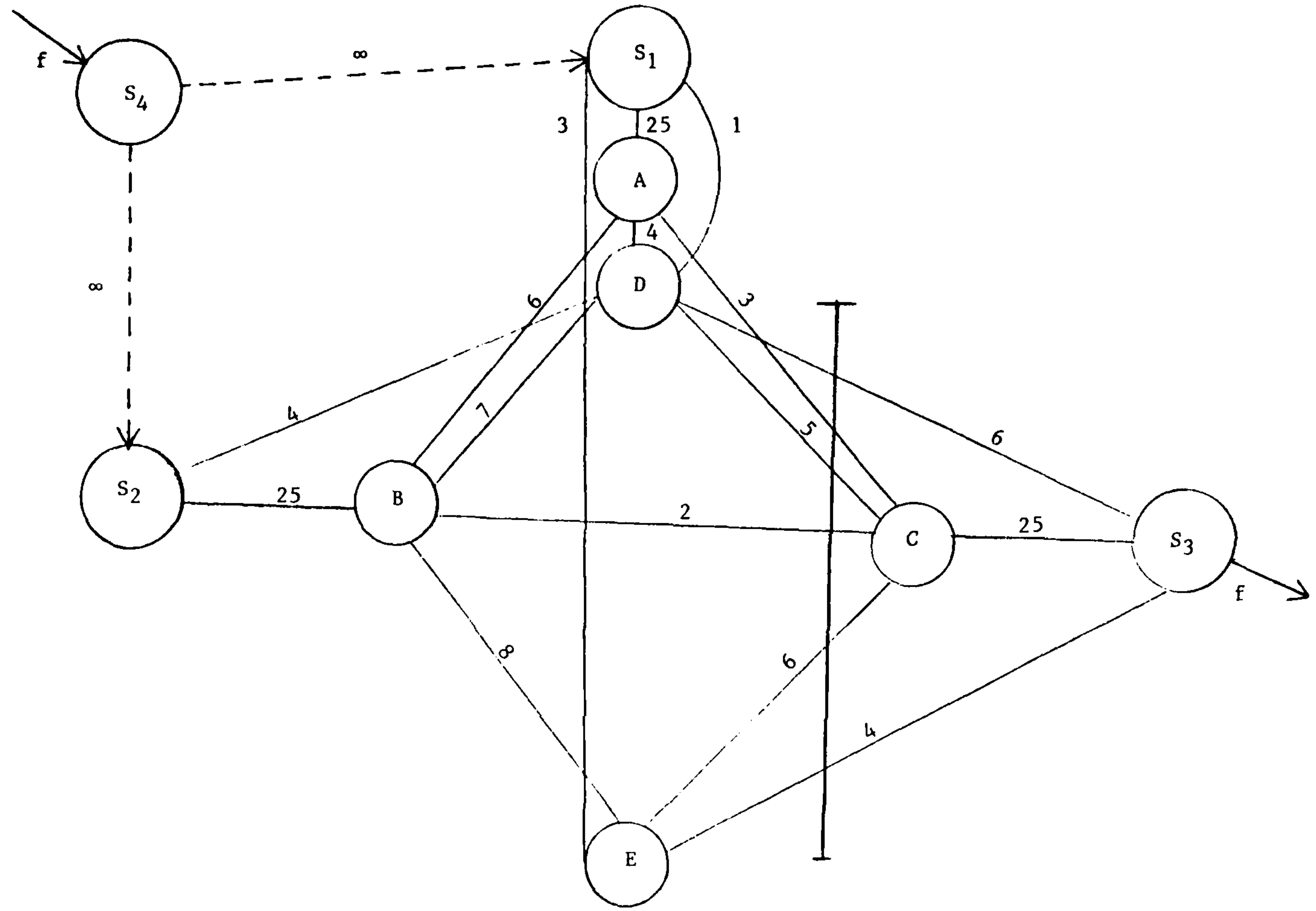


Figure 3.11 Illustration of processor S_3 active and the dummy processor S_4 connected to S_1 and S_2.

The same was done to the other processors. This time S_2 was active and dummy processor S_4 was connected to S_1 and S_3 respectively. The cut-set in this case was 27 and included branches AB, CB, DB, EB and DS_2 (See Fig. 3.10 and computer print-out in Appendix IV-B).

When S_3 was active and dummy processor S_4 was connected to S_1 and S_2 respectively, the cut-set was 26. The branches include AC, BC, DC, EC, DS_3 and ES_3 (See Fig. 3.11 and computer print-out in Appendix IV-C).

Now consider again the example of three-processor and five module network shown in Fig. 3.8, with the running times for each processor given in Table VII.

The linear programming model was used to determine the cut-set when all the three processors were active. The value of the objective function was found to be 38, which gave the following cut-set (See Fig. 3.12).

In this cut-set the relationship between Fig. 3.9 and Fig. 3.10 is shown, where the cut-set values were 17 and 27 respectively. The total of these two cut-sets equals 44. Subtracting the common arc which is AB with a processing time of 6 yields 38, the present cut-set value. Similarly from Fig 3.9 the cut-set value of 7 minus the arc AB of 6 gives a value of 11, which when combined with Fig. 3.10 whose cut-set value of 27 gives the present cut-set value of 38.

Considering the above factors and the distribution of modules to processors in Fig. 3.12 the following assignment is the solution,

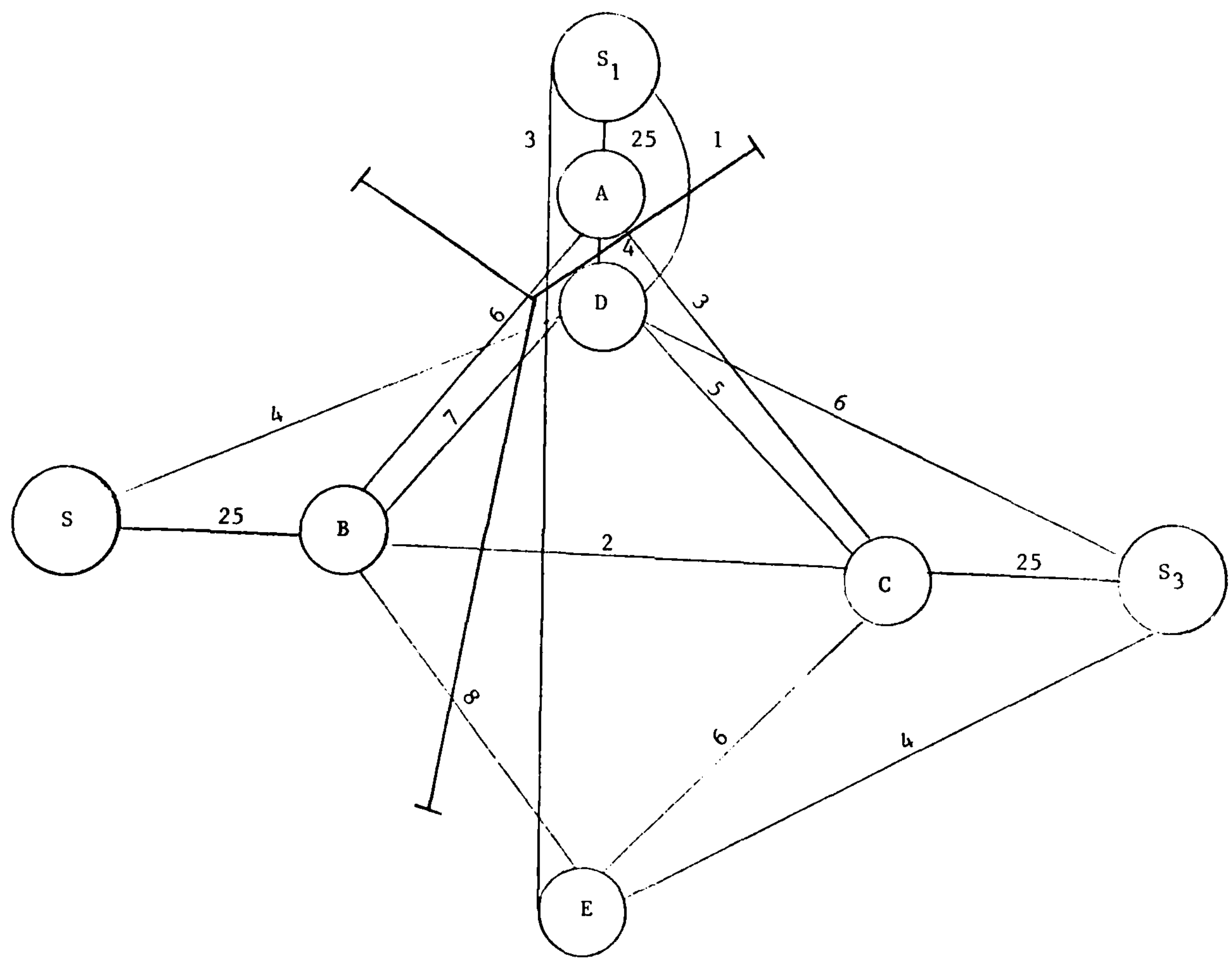


Figure 3.12 Optimal module to processor schedule.

Module A to Processor S_1

Module B to Processor S_2

Module C to Processor S_3

Module D to Processor S_3

Module E to Processor S_3

The linear programming model constructed was used with the LINDO[9] software package to get the above assignment solution (See Appendix IV).

4.0 SCHEDULING ALGORITHM

4.1 Multiprocessor Scheduling Algorithm

Assumptions:

The following are the assumptions in connection with the developemnt of the model.

1. The algorithm assumes that each module sends its received message forward, along a chosen link to the processor. This means that when a module has to be processed it does to the processor along a link for best execution time.

2. The processors in the network are homogeneous and they are fully connected. This means that all the modules in the network are fully connected to the processors. Except for any attached modules which must be processed by a particular processor, in general the modules could go to any processor for processing.

3. The message eventually arrive at their final destination, taking into consideration the fastest computation time on any given processor. This means that, when a module has to be processed, it should be processed by the processor which requires the minimum amount of time.

4. The module-to-processor communication time is known and given.

5. The number of modules in a network is, in general, much higher than the number of processors.

It is noted in this connection that the efficiency goes down by increasing the number of processors, and hence only three processors, an 80% efficiency level has been used. This observation was based on research done at Digital on 2 DEC Systems, verifying that when additional processors are included after the third procesor to the system, the efficiency of the system goes down [2].

The general procedure for the proposed heuristic in now defined.

<u>STEP 1.</u>

Establish the computational time constant. This constraint assumes that a module will be assigned, only to the processor which takes the least amount of time for processing, if it is not an attached module.

<u>STEP 2.</u>

Establish the load balancing constraint. In our case we assume that a particular processor at any given time should not process more than 10 Million instructions/sec and less than 0.5 Million instruction/sec, that is 0.5 Million < Information/sec<10 Million. In the first case it would be considered to be overloaded and in the latter case it would be considered to be underloaded.

STEP 3.

Establish the priority constraint. In some cases in a network, some modules may have to be assigned to one particular processor in order to be processed, taking into consideration their unique capabilities. In this respect we could first assign those modules to the particular processor and then start from Step 4.

STEP 4.

Set up the precedence relationships among the module to be processed. This will identify which module should be processed first for a given program or operation.

STEP 5.

For each program or operation, create a set of I number of modules through which that program has to pass, to be processed. Set up the precedence relationship(s) for each program or operation using the relationship(s) established in Step 4.

STEP 6.

In relation to the overall performance of the procedure discussed for this problem, one should consider the utilization rate of the functions. This means finding out which modules in the network occurs most. Then the modules are to be set accordingly, to carry out those particular functions. Also, it is required to utilize as many modules as possible to handle the most frequent functions.

STEP 7.

It is assumed that all the available processors have the capability to process each of the above modules which occur frequently can be assigned to any available processor for the fastest computation time.

STEP 8.

Assign the set of the I modules formed in Step 5 to processors so that those modules could be processed fastest. Take computation time on a given processor into consideration.

STEP 9.

Make the above assignment to separate processors so that the load balancing is maximized, without violating the precedence relationships.

STEP 10.

Choose the mth assignment (corresponding to $I = m$) from Step 8, and if this assignment satisfies the load balancing constraint and the computation time constraint along with the attached module constraint (if any), then stop; otherwise go to Step 11.

STEP 11.

Identify the overloaded and underloaded processors.

STEP 12.

Assign some modules from overloaded processors to underloaded processors taking Steps 1, 2, and 3 into consideration. Then go to Step 10.

The algorithm presented here provides a scheduling scheme for the assignment of different modules to available processors, to minimize the total time on a given processor, on a distributed system.

5.0 SUMMARY

The problem of finding an assignment of modules to processors to minimize the total processing time, where the module-to -processor communication time is given, has been considered. The algorithm presented in this report provides a scheme for assignment of modules to processor in a distributed computer system.

The model represents modules as nodes in a graph and the module-to-processor communication times as weights on the undirected arcs connecting modules to processors. The methodology is to assign modules to processors so that total processing time is minimized. The standard max-flow- min-cut algorithm [8] can obtain the required assignment. However, the computation becomes intractable for a large number of processors.

The proposed linear programming model minimizes total processing time subject to resource limitation constraints described in Section 4.0. Moreover, a procedure has been shown for handling additional modules introduced to the network, taking the utilization rate into consideration. The utilization rate of each function which is nothing but a function of each module, can be found simply by executing a test run before the initial setting up of the computer systems. This could also be updated periodically.

The heuristic models provide appropriate solutions to the module assignment problem. The basic idea of heuristic methods is to find an assignment for the modules to processors and to balance the load among the processors. Load balancing is an important problem in distributed computer systems, a problem which is yet far from being solved completely. Also, the question remains how the relationship will be expressed between the precedence relations and the queuing delay. These are definitely important problems requiring further research.

6.0 REFERENCES

1. V. Balanchandsan, J. W. McCredie and O. I. Miklait, "Models of the Job Allocation Problem in Computer Networks", Proc. Compcon, Fall 1973, pp. 211-214.

2. Digital Introduction to DEC-net, Massachusetts, January 1980.

3. Edward K. Bowdon, Sr., "Priority Assignment in a Network of Computers", IEEE Transactions on Computers, Vol. C-18, No. 11, November 1969.

4. V. B. Gylys and J. A. Edwards, "Optimal Partitioning of Workload for Distributed Systems", Tutorial of IEEE Catalog, No. EH0151-1, 1979.

5. Harold S. Stone, "Multiprocessor Scheduling With the Air of Network Flow Algorithms", Tutorial of IEEE Catalog, No. EH0151-1, 1979.

6. Honey E. Elovitz and Constance L.Heitmeyer, "What Is A Computer Network", IEEE 1974 NTC Record, pp. 1007-1014, 1974.

7. Kemal Efe, "Heuristic Models of Task Assignment Scheduling in Distributed Systems", IEEE Computer, Vol. June 1982, pp. 50-56.

8. Mokhtam S. Bazarra and John J. Jarvis, "Linear Programming and Network Flows", John Wiley, New York, 1977.

9. Schrage Linus, "Users Manual for LINDO", University of Chicago, Chicago, Illinois , December 1981.

10. Wesley W. Chu, Leslie J. Holloway, Min-Isung Lan, and Kemal Efe, "Task Allocation in Distributed Data Processing", Computer Vol. 13, No. 11, November 7, 1980, pp. 57-69.

11. Edirsinghe, Janaka, A Heuristic Approach For Module Scheduling In A Distributed Computer System, M. S. Thesis, North Carolina A& T State University, October 1982.

12. Pulat, S. and Janaka Edirisinghe, "Network Flow Approach to Multiprocessor Scheduling", Presented at the 1982 Joint National Manufacturing of ORSA/TIMS at San Diego.

13. Frederick, M. and Harold Martin, "Reliability Method For Maximum Efficiency in a System of Interconnected Processors", To be presented at the 15th Southeastern Symposium on System Theory, March 1983.

APPENDICES

APPENDIX I

SOLUTION OF THE ASSIGNMENT PROBLEM

```
..R LINDO
  LINDO  (UC4APR79)
:*RETR
FILE NAME=
*GAMINI
:*LOOK
ROW
:*ALL
MIN     3 XA1 + 10 XB1 + 12 XC1 + 10 XD1 + 4 XE1 + 12 XA2 + 4 XB2
     + 14 XC2 + 7 XD2 + 7 XE2 + 15 XA3 + 12 XB3 + 4 XC3 + 5 XD3 + 3 XE3
     +  WAB1 +  WAB2 +  WAB3 +  YAB1 +  YAB2 +  YAB3 + 1.5 WAC1
     + 1.5 WAC2 + 1.5 WAC3 + 1.5 YAC1 + 1.5 YAC2 + 1.5 YAC3 + 1.5 WAD1
     + 1.5 WAD2 + 1.5 WAD3 + 1.5 YAD1 + 1.5 YAD2 + 1.5 YAD3 + 1.5 WBC1
     + 1.5 WBC2 + 1.5 WBC3 + 1.5 YBC1 + 1.5 YBC2 + 1.5 YBC3 +  WBD1
     +  WBD2 +  WBD3 +  YBD1 +  YBD2 +  YBD3 + 0.5 WBE1 + 0.5 WBE2
     + 0.5 WBE3 + 0.5 YBE1 + 0.5 YBE2 + 0.5 YBE3 +  WCD1 +  WCD2 +  WCD3
     +  YCD1 +  YCD2 +  YCD3 + 3.5 WCE1 + 3.5 WCE2 + 3.5 WCE3 + 3.5 YCE1
     + 3.5 YCE2 + 3.5 YCE3
SUBJECT TO
    2)      XA1 +  XA2 +  XA3 =     1
    3)      XB1 +  XB2 +  XB3 =     1
    4)      XC1 +  XC2 +  XC3 =     1
    5)      XD1 +  XD2 +  XD3 =     1
    6)      XE1 +  XE2 +  XE3 =     1
    7)      XA1 -  XB1 +  WAB1 -  YAB1 =     0
    8)      XA2 -  XB2 +  WAB2 -  YAB2 =     0
    9)      XA3 -  XB3 +  WAB3 -  YAB3 =     0
   10)      XA1 -  XC1 +  WAC1 -  YAC1 =     0
   11)      XA2 -  XC2 +  WAC2 -  YAC2 =     0
   12)      XA3 -  XC3 +  WAC3 -  YAC3 =     0
   13)      XA1 -  XD1 +  WAD1 -  YAD1 =     0
   14)      XA2 -  XD2 +  WAD2 -  YAD2 =     0
   15)      XA3 -  XD3 +  WAD3 -  YAD3 =     0
   16)      XB1 -  XC1 +  WBC1 -  YBC1 =     0
   51)      XB2 -  XC2 +  WBC2 -  YBC2 =     0
   17)      XB3 -  XC3 +  WBC3 -  YBC3 =     0
   18)      XB1 -  XD1 +  WBD1 -  YBD1 =     0
   19)      XB2 -  XD2 +  WBD2 -  YBD2 =     0
   20)      XB3 -  XD3 +  WBD3 -  YBD3 =     0
   21)      XB1 -  XE1 +  WBE1 -  YBE1 =     0
   22)      XB2 -  XE2 +  WBE2 -  YBE2 =     0
   23)      XB3 -  XE3 +  WBE3 -  YBE3 =     0
   24)      XC1 -  XD1 +  WCD1 -  YCD1 =     0
   25)      XC2 -  XD2 +  WCD2 -  YCD2 =     0
   26)      XC3 -  XD3 +  WCD3 -  YCD3 =     0
   27)      XC1 -  XE1 +  WCE1 -  YCE1 =     0
   28)      XC2 -  XE2 +  WCE2 -  YCE2 =     0
   47)      XC3 -  XE3 +  WCE3 -  YCE3 =     0
   29)      XD1 -  XE1 +  WDE1 -  YDE1 =     0
   30)      XD2 -  XE2 +  WDE2 -  YDE2 =     0
   31)      XD3 -  XE3 +  WDE3 -  YDE3 =     0
   32)      XA1  <=    1
   33)      XB1  <=    1
   34)      XC1  <=    1
   35)      XD1  <=    1
```

```
  36)      XE1 <=   1
  37)      XA2 <=   1
  38)      XB2 <=   1
  39)      XC2 <=   1
  40)      XD2 <=   1
  41)      XE2 <=   1
  42)      XA3 <=   1
  43)      XB3 <=   1
  44)      XC3 <=   1
  45)      XD3 <=   1
  46)      XE3 <=   1
  48)      XA1 + XB1 + XC1 + XD1 + XE1 >=   1
  49)      XA2 + XB2 + XC2 + XD2 + XE2 >=   1
  50)      XA3 + XB3 + XC3 + XD3 + XE3 >=   1
END
   NUMBER INTEGER VARIABLES=  15
:*GO
    LP OPTIMUM FOUND  AT STEP    44
         OBJECTIVE FUNCTION VALUE
 1)          33.00000
VARIABLE            VALUE            REDUCED COST
 XA1              1.000000           3.000000
 XB1              0.000000           0.000000
 XC1              0.000000           0.000000
 XD1              0.000000           0.000000
 XE1              0.000000           0.000000
 XA2              0.000000           0.000000
 XB2              1.000000           4.000000
 XC2              0.000000           1.000000
 XD2              0.000000           0.000000
 XE2              0.000000           0.000000
 XA3              0.000000           2.000000
 XB3              0.000000           0.000000
 XC3              1.000000           0.000000
 XD3              1.000000           0.000000
 XE3              1.000000           0.000000
 WAB1             0.000000           2.000000
 WAB2             1.000000           0.000000
 WAB3             0.000000           0.000000
 YAB1             1.000000           0.000000
 YAB2             0.000000           2.000000
 YAB3             0.000000           2.000000
 WAC1             0.000000           3.000000
 WAC2             0.000000           3.000000
 WAC3             1.000000           0.000000
 YAC1             1.000000           0.000000
 YAC2             0.000000           0.000000
 YAC3             0.000000           3.000000
 WAD1             0.000000           3.000000
 WAD2             0.000000           0.000000
 WAD3             1.000000           0.000000
 YAD1             1.000000           0.000000
 YAD2             0.000000           3.000000
 YAD3             0.000000           3.000000
 WBC1             0.000000           3.000000
```

WBC2	0.000000	3.000000
WBC3	1.000000	0.000000
YBC1	0.000000	0.000000
YBC2	1.000000	0.000000
YBC3	0.000000	3.000000
WBD1	0.000000	2.000000
WBD2	0.000000	2.000000
WBD3	1.000000	0.000000
YBD1	0.000000	0.000000
YBD2	1.000000	0.000000
YBD3	0.000000	2.000000
WBE1	0.000000	0.000000
WBE2	0.000000	1.000000
WBE3	1.000000	0.000000
YBE1	0.000000	1.000000
YBE2	1.000000	0.000000
YBE3	0.000000	1.000000
WCD1	0.000000	1.000000
WCD2	0.000000	0.000000
WCD3	0.000000	2.000000
YCD1	0.000000	1.000000
YCD2	0.000000	2.000000
YCD3	0.000000	0.000000
WCE1	0.000000	0.000000
WCE2	0.000000	1.000000
WCE3	0.000000	0.000000
YCE1	0.000000	7.000000
YCE2	0.000000	6.000000
YCE3	0.000000	7.000000
WDE1	0.000000	0.000000
YDE1	0.000000	0.000000
WDE2	0.000000	0.000000
YDE2	0.000000	0.000000
WDE3	0.000000	0.000000
YDE3	0.000000	0.000000

ROW	SLACK	DUAL PRICES
2)	0.000000	-9.000000
3)	0.000000	-10.000000
4)	0.000000	-4.500000
5)	0.000000	-5.500000
6)	0.000000	-7.000000
7)	0.000000	1.000000
8)	0.000000	-1.000000
9)	0.000000	-1.000000
10)	0.000000	1.500000
11)	0.000000	1.500000
12)	0.000000	-1.500000
13)	0.000000	1.500000
14)	0.000000	-1.500000
15)	0.000000	-1.500000
16)	0.000000	1.500000
17)	0.000000	-1.500000
18)	0.000000	1.000000
19)	0.000000	1.000000
20)	0.000000	-1.000000

```
21)          0.000000       -0.500000
22)          0.000000        0.500000
23)          0.000000       -0.500000
24)          0.000000        0.000000
25)          0.000000       -1.000000
26)          0.000000        1.000000
27)          0.000000       -3.500000
28)          0.000000       -2.500000
29)          0.000000        0.000000
30)          0.000000        0.000000
31)          0.000000        0.000000
32)          0.000000        0.000000
33)          1.000000        0.000000
34)          1.000000        0.000000
35)          1.000000        0.000000
36)          1.000000        0.000000
37)          1.000000        0.000000
38)          0.000000        0.000000
39)          1.000000        0.000000
40)          1.000000        0.000000
41)          1.000000        0.000000
42)          1.000000        0.000000
43)          1.000000        0.000000
44)          0.000000        0.000000
45)          0.000000        0.000000
46)          0.000000        0.000000
47)          0.000000       -3.500000
48)          0.000000       -1.000000
49)          0.000000       -2.000000
50)          2.000000        0.000000
51)          0.000000        1.500000
NO. ITERATIONS=      44
 BRANCHES=    0 DETERM.=  4.000E  0
 DO RANGE(SENSITIVITY) ANALYSIS?
:*NO
LP OPTIMUM IS IP OPTIMUM
:*QUIT
STOP
```

APPENDIX II

ASSIGNMENT OF EXTRA MODULES TO PROCESSORS

```
MIN       3 XA1 + 12 XA2 + 15 XA3 + 10 XB1 + 4 XB2 + 12 XB3 + 12 XC1
     + 14 XC2 + 4 XC3 + 10 XD1 + 7 XD2 + 5 XD3 + 4 XE1 + 7 XE2 + 3 XE3
SUBJECT TO
    2)      XA1 +  XA2 +  XA3 =      0
    3)      XB1 +  XB2 +  XB3 =      1
    4)      XC1 +  XC2 +  XC3 =      2
    5)      XD1 +  XD2 +  XD3 =      1
    6)      XE1 +  XE2 +  XE3 =      2
    7)      XA1 =     0
    8)      XB2 =     0
    9)      XC3 =     0
   10)      XD3 =     0
   11)      XE3 =     0
   12)      XA1 <=    1
   13)      XA2 <=    1
   14)      XA3 <=    1
   15)      XB1 <=    1
   16)      XB2 <=    1
   17)      XC1 <=    1
   18)      XB3 <=    1
   19)      XC2 <=    1
   20)      XC3 <=    1
   21)      XD1 <=    1
   22)      XD2 <=    1
   23)      XD3 <=    1
   24)      XE1 <=    1
   25)      XE2 <=    1
   26)      XE3 <=    1
END

GO

     LP OPTIMUM FOUND  AT STEP     10

          OBJECTIVE FUNCTION VALUE

  1)          54.00000

VARIABLE          VALUE           REDUCED COST
 XA1            0.000000           3.000000
 XA2            0.000000          12.000000
 XA3            0.000000          15.000000
 XB1            1.000000           0.000000
 XB2            0.000000           0.000000
 XB3            0.000000           2.000000
 XC1            1.000000           0.000000
 XC2            1.000000           0.000000
 XC3            0.000000           0.000000
 XD1            0.000000           3.000000
 XD2            1.000000           0.000000
 XD3            0.000000           0.000000
 XE1            1.000000           0.000000
 XE2            1.000000           0.000000
 XE3            0.000000           0.000000
```

ROW	SLACK	DUAL PRICES
2)	0.000000	0.000000
3)	0.000000	10.000000
4)	0.000000	14.000000
5)	0.000000	-7.000000
6)	0.000000	-7.000000
7)	0.000000	0.000000
8)	0.000000	6.000000
9)	0.000000	10.000000
10)	0.000000	2.000000
11)	0.000000	4.000000
12)	1.000000	0.000000
13)	1.000000	0.000000
14)	1.000000	0.000000
15)	0.000000	0.000000
16)	1.000000	0.000000
17)	0.000000	2.000000
18)	1.000000	0.000000
19)	0.000000	0.000000
20)	1.000000	0.000000
21)	1.000000	0.000000
22)	0.000000	0.000000
23)	1.000000	0.000000
24)	0.000000	3.000000
25)	0.000000	0.000000
26)	1.000000	0.000000

NO. ITERATIONS= 10

DO RANGE(SENSITIVITY) ANALYSIS?
:N

APPENDIX III

NETWORK FLOW APPROACH TO TWO PROCESSOR SCHEDULING

EXAMPLE PROBLEM

```
MAX        F
SUBJECT TO
    2)     F -  FS1A -  FS1B -  FS1C -  FS1D -  FS1E -  FS1F =     0
    3)     FS1A +  FBA +  FFA +  FCA -  FAB -  FAF -  FAC -  FAS2 =     0
    4)     FS1B -  FBA +  FAB +  FEB +  FDB -  FBS2 -  FBE -  FBD =     0
    5)     FS1C -  FCA +  FAC +  FEC -  FCS2 -  FCE =    0
    6)     FS1D -  FDB +  FBD +  FED -  FDS2 -  FDE =    0
    7)     FS1E -  FEB +  FBE -  FEC +  FCE -  FED +  FDE -  FES2 =     0
    8)     FS1F -  FFA +  FAF -  FFS2 =    0
    9)  -  F +  FAS2 +  FBS2 +  FCS2 +  FDS2 +  FES2 +  FFS2 =     0
   10)     FS1A <=    10
   11)     FS1B <=    10
   12)     FS1C <=    4
   13)     FS1D <=    3
   14)     FS1E <=  . 12
   15)     FS1F <=    4
   16)     FAB  <=   9
   17)     FBA  <=   9
   18)     FAC  <=   4
   19)     FCA  <=   4
   20)     FAF  <=   12
   21)     FFA  <=   12
   22)     FBD  <=   12
   23)     FDB  <=   12
   24)     FBE  <=   3
   25)     FEB  <=   3
   26)     FCE  <=   11
   27)     FEC  <=   11
   28)     FDE  <=   5
   29)     FED  <=   5
   30)     FAS2  <=   5
   31)     FBS2  <=   2
   32)     FCS2  <=   4
   33)     FDS2  <=   6
   34)     FES2  <=   5
   35)     FFS2  <=   18
END

:GO

     LP OPTIMUM FOUND  AT STEP     17

          OBJECTIVE FUNCTION VALUE

  1)          38.00000
```

VARIABLE	VALUE	REDUCED COST
F	38.000000	0.000000
FS1A	10.000000	0.000000
FS1B	9.000000	0.000000
FS1C	4.000000	0.000000
FS1D	3.000000	0.000000
FS1E	8.000000	0.000000
FS1F	4.000000	0.000000
FBA	7.000000	0.000000
FFA	0.000000	1.000000
FCA	0.000000	0.000000
FAB	0.000000	0.000000
FAF	12.000000	0.000000
FAC	0.000000	0.000000
FAS2	5.000000	0.000000
FEB	0.000000	0.000000
FDB	0.000000	0.000000
FBS2	2.000000	0.000000
FBE	0.000000	0.000000
FBD	0.000000	0.000000
FEC	0.000000	0.000000
FCS2	4.000000	0.000000
FCE	0.000000	0.000000
FED	3.000000	0.000000
FDS2	6.000000	0.000000
FDE	0.000000	0.000000
FES2	5.000000	0.000000
FFS2	16.000000	0.000000

ROW	SLACK	DUAL PRICES
2)	0.000000	1.000000
3)	0.000000	1.000000
4)	0.000000	1.000000
5)	0.000000	1.000000
6)	0.000000	1.000000
7)	0.000000	1.000000
8)	0.000000	0.000000
9)	0.000000	0.000000
10)	0.000000	0.000000
11)	1.000000	0.000000
12)	0.000000	0.000000
13)	0.000000	0.000000
14)	4.000000	0.000000
15)	0.000000	1.000000
16)	9.000000	0.000000
17)	2.000000	0.000000
18)	4.000000	0.000000
19)	4.000000	0.000000
20)	0.000000	1.000000
21)	12.000000	0.000000
22)	12.000000	0.000000
23)	12.000000	0.000000

```
24)            3.000000          0.000000
25)            3.000000          0.000000
26)           11.000000          0.000000
27)           11.000000          0.000000
28)            5.000000          0.000000
29)            2.000000          0.000000
30)            0.000000          1.000000
31)            0.000000          1.000000
32)            0.000000          1.000000
33)            0.000000          1.000000
34)            0.000000          1.000000
35)            2.000000          0.000000

NO. ITERATIONS=       17

 DO RANGE(SENSITIVITY) ANALYSIS?
>N

:QUIT
```

APPENDIX IV-A

'JAN'; PROCESSOR S_1 IS ACTIVE AND THE DUMMY PROCESSOR

S_4 IS CONNECTED TO PROCESSOR S_2 & S_3

APPENDIX IV

NETWORK FLOW APPROACH TO THREE PROCESSOR PROBLEM

APPENDIX IV-A

'JAN'; PROCESSOR S_1 IS ACTIVE AND THE DUMMY PROCESSOR S_4 IS CONNECTED TO PROCESSOR S_2 & S_3

```
..R LINDOS
  LINDO  (UC3AUG79)
:*RETR
FILE NAME=
*JAN
:*LOOK
ROW
>*ALL
MAX      F
SUBJECT TO
    2)       FAS1 +  FAB +  FAC +  FAD -  FBA -  FCA -  FDA =     0
    3)   -  FAB +  FBA +  FBC +  FBD +  FBE +  FBS2 -  FCB -  FDB -  FEB
      -  FS2B =    0
    4)   -  FAC +  FCA -  FBC +  FCB +  FCD +  FCE +  FCS3 -  FDC -  FEC
      -  FS3C =    0
    5)   -  FAD +  FDA -  FCD +  FDC +  FDS1 +  FDS2 +  FDS3 -  FS2D -  FS3D
      =    0
    6)   -  FBE +  FEB -  FCE +  FEC +  FES1 +  FES3 -  FS3E =     0
    7)       F -  FAS1 -  FDS1 -  FES1 =     0
    8)   -  FBS2 +  FS2B -  FDS2 +  FS2D -  FS4S2 =     0
    9)   -  FCS3 +  FS3C -  FDS3 +  FS3D -  FES3 +  FS3E -  FS4S3 =     0
   10)   -  F +  FS4S2 +  FS4S3 =     0
   11)       FAS1 <=   1000
   12)       FES1 <=   3
   13)       FDS1 <=   1
   14)       FS4S2 <=   1000
   15)       FS4S3 <=   1000
   16)       FAB <=   6
   17)       FBA <=   6
   18)       FAD <=   4
   19)       FDA <=   4
   20)       FAC <=   3
   21)       FCA <=   3
   22)       FDS2 <=   4
   23)       FS2D <=   4
   24)       FDB <=   7
   25)       FBD <=   7
   26)       FDC <=   5
   27)       FCD <=   5
   28)       FDS3 <=   6
   29)       FS3D <=   6
   30)       FCB <=   2
   31)       FBC <=   2
   32)       FCS3 <=   1000
   33)       FS3C <=   1000
   34)       FCE <=   6
   35)       FEC <=   6
   36)       FBE <=   8
   37)       FEB <=   3
   38)       FES3 <=   4
   39)       FS3E <=   4
   40)       FBS2 <=   1000
   41)       FS2B <=   1000
END
:*GO
```

LP OPTIMUM FOUND AT STEP 17

OBJECTIVE FUNCTION VALUE

1) 17.00000

VARIABLE	VALUE	REDUCED COST
F	17.000000	0.000000
FAS1	13.000000	0.000000
FA3	0.000000	1.000000
FAC	0.000000	1.000000
FAD	0.000000	1.000000
FBA	6.000000	0.000000
FCA	3.000000	0.000000
FDA	4.000000	0.000000
FBC	2.000000	0.000000
FBD	0.000000	0.000000
FBE	0.000000	0.000000
FBS2	0.000000	0.000000
FCB	0.000000	0.000000
FDB	0.000000	0.000000
FEB	0.000000	0.000000
FS23	8.000000	0.000000
FCD	0.000000	0.000000
FCE	0.000000	0.000000
FCS3	0.000000	0.000000
FDC	0.000000	0.000000
FEC	0.000000	0.000000
FS3C	1.000000	0.000000
FDS1	1.000000	0.000000
FDS2	0.000000	0.000000
FDS3	0.000000	0.000000
FS2D	4.000000	0.000000
FS3D	1.000000	0.000000
FES1	3.000000	0.000000
FES3	0.000000	0.000000
FS3E	3.000000	0.000000
FS4S2	12.000000	0.000000
FS4S3	5.000000	0.000000

ROW	SLACK	DUAL PRICES
2)	0.000000	1.000000
3)	0.000000	0.000000
4)	0.000000	0.000000
5)	0.000000	0.000000
6)	0.000000	0.000000
7)	0.000000	1.000000
8)	0.000000	0.000000
9)	0.000000	0.000000
10)	0.000000	0.000000
11)	987.000000	0.000000
12)	0.000000	1.000000
13)	0.000000	1.000000
14)	363.000000	0.000000
15)	395.000000	0.000000
16)	0.000000	0.000000
17)	0.000000	1.000000
18)	4.000000	0.000000
19)	0.000000	1.000000

```
20)                3.000000          0.000000
21)                0.000000          1.000000
22)                4.000000          0.000000
23)                0.000000          0.000000
24)                7.000000          0.000000
25)                7.000000          0.000000
26)                5.000000          0.000000
27)                5.000000          0.000000
28)                6.000000          0.000000
29)                5.000000          0.000000
30)                2.000000          0.000000
31)                0.000000          0.000000
32)             1000.000000          0.000000
33)              999.000000          0.000000
34)                6.000000          0.000000
35)                6.000000          0.000000
36)                8.000000          0.000000
37)                3.000000          0.000000
38)                4.000000          0.000000
39)                1.000000          0.000000
40)             1000.000000          0.000000
41)              992.000000          0.000000
NO. ITERATIONS=        17
 DO RANGE(SENSITIVITY) ANALYSIS?
>*NO
:*QUIT
STOP
```

APPENDIX IV-B

"JANA'; PROCESSOR S_2 IS ACTIVE AND THE DUMMY PROCESSOR

S_4 IS CONNECTED TO PROCESSOR S_1 & S_3

```
..R LINDOS
  LINDO  (UC3AUG79)
:*RETR
FILE NAME=
*JANA
:*LOOK
ROW
>*ALL
MAX      F
SUBJECT TO
   2)     FAS1 +  FAB +  FAC +  FAD -  FS1A -  FBA -  FCA -  FDA =
   3)  -  FAB +  FBA +  FBS2 +  FBC +  FBD +  FBE -  FCB -  FDB -  FEB
    =     0
   4)  -  FAC +  FCA -  FBC +  FCB +  FCD +  FCE +  FCS3 -  FDC -  FEC
    -  FS3C =    0
   5)  -  FAD +  FDA -  FBD +  FDB -  FCD +  FDC +  FDS2 +  FDS3 -  FDE
    -  FS3D -  FS1D =    0
   6)  -  FBE +  FEB -  FCE +  FEC +  FES1 +  FES3 -  FS1E -  FS3E =
   7)  -  FAS1 +  FS1A -  FDS1 +  FS1D -  FES1 +  FS1E -  FS4S1 =    0
   8)     F -  FBS2 -  FDS2 =    0
   9)  -  FCS3 +  FS3C -  FDS3 +  FS3D -  FES3 +  FS3E -  FS4S3 =    0
  10)  -  F +  FS4S1 +  FS4S3 =    0
  11)     FAS1 <=   25
  12)     FS1A <=   25
  13)     FAD <=   4
  14)     FDA <=   4
  15)     FAB <=   6
  16)     FBA <=   6
  17)     FAC <=   3
  18)     FCA <=   3
  19)     FBC <=   2
  20)     FCB <=   2
  21)     FBD <=   7
  22)     FDB <=   7
  23)     FBE <=   8
  24)     FEB <=   8
  25)     FCD <=   5
  26)     FDC <=   5
  27)     FCE <=   6
  28)     FEC <=   6
  29)     FCS3 <=   25
  30)     FS3C <=   25
  31)     FDS1 <=   1
  32)     FS1D <=   1
  33)     FDS3 <=   6
  34)     FS3D <=   6
  35)     FES4 <=   4
  36)     FS3E <=   4
  37)     FES1 <=   3
  38)     FS1E <=   3
  39)     FBS2 <=   25
  40)     FDS2 <=   4
  41)     FS4S1 <=   1000
  42)     FS4S3 <=   1000
  43)     FES3 <=   4
```

```
END
:*GO
    LP OPTIMUM FOUND  AT STEP     20
         OBJECTIVE FUNCTION VALUE
 1)            27.00000
VARIABLE         VALUE           REDUCED COST
 F             27.000000           0.000000
 FAS1           0.000000           0.000000
 FAB            6.000000           0.000000
 FAC            3.000000           0.000000
 FAD            4.000000           0.000000
 FS1A          13.000000           0.000000
 FBA            0.000000           1.000000
 FCA            0.000000           0.000000
 FDA            0.000000           0.000000
 FBS2          23.000000           0.000000
 FBC            0.000000           1.000000
 FBD            0.000000           1.000000
 FBE            0.000000           1.000000
 FCB            2.000000           0.000000
 FDB            7.000000           0.000000
 FEB            8.000000           0.000000
 FCD            0.000000           0.000000
 FCE            1.000000           0.000000
 FCS3           0.000000           0.000000
 FDC            0.000000           0.000000
 FEC            0.000000           0.000000
 FS3C           0.000000           0.000000
 FDS2           4.000000           0.000000
 FDS3           0.000000           0.000000
 FDS1           0.000000           0.000000
 FS3D           6.000000           0.000000
 FS1D           1.000000           0.000000
 FES1           0.000000           0.000000
 FES3           0.000000           0.000000
 FS1E           3.000000           0.000000
 FS3E           4.000000           0.000000
 FS4S1         17.000000           0.000000
 FS4S3         10.000000           0.000000
 FES4           0.000000           0.000000
ROW              SLACK            DUAL PRICES
 2)             0.000000           0.000000
 3)             0.000000           1.000000
 4)             0.000000           0.000000
 5)             0.000000           0.000000
 6)             0.000000           0.000000
 7)             0.000000           0.000000
 8)             0.000000           1.000000
 9)             0.000000           0.000000
10)             0.000000           0.000000
11)            25.000000           0.000000
12)            12.000000           0.000000
13)             0.000000           0.000000
14)             4.000000           0.000000
15)             0.000000           1.000000
```

```
16)          6.000000          0.000000
17)          0.000000          0.000000
18)          3.000000          0.000000
19)          2.000000          0.000000
20)          0.000000          1.000000
21)          7.000000          0.000000
22)          0.000000          1.000000
23)          3.000000          0.000000
24)          0.000000          1.000000
25)          5.000000          0.000000
26)          5.000000          0.000000
27)          5.000000          0.000000
28)          6.000000          0.000000
29)         25.000000          0.000000
30)         25.000000          0.000000
31)          1.000000          0.000000
32)          0.000000          0.000000
33)          6.000000          0.000000
34)          0.000000          0.000000
35)          4.000000          0.000000
36)          0.000000          0.000000
37)          3.000000          0.000000
38)          0.000000          0.000000
39)          2.000000          0.000000
40)          0.000000          1.000000
41)        983.000000          0.000000
42)        990.000000          0.000000
43)          4.000000          0.000000
NO. ITERATIONS=       20
 DO RANGE(SENSITIVITY) ANALYSIS?
>*NO
:*QUIT
STOP
```

APPENDIX IV-C

'JANAK'; PROCESSOR S_3 IS ACTIVE AND THE DUMMY PROCESSOR S_4 IS CONNECTED TO PROCESSOR S_1 & S_2

```
 LINDO  (UC3AUG79)
:*RETR

FILE NAME=
*JANAK

:*LOOK

ROW
>*ALL


MAX      F
SUBJECT TO
   2)     FAB +  FAC +  FAD +  FAS1 -  FBA -  FCA -  FDA -  FS1A =    0
   3)  -  FAB +  FBA +  FBC +  FBD +  FBE +  FBS2 -  FCB -  FDB -  FEB
     -  FS2B =    0
   4)  -  FAC +  FCA -  FBC +  FCB +  FCD +  FCE +  FCS3 -  FDC -  FEC
     =    0
   5)  -  FAD +  FDA -  FBD +  FDB -  FCD +  FDC +  FDS1 +  FDS2 +  FDS3
     -  FS1D -  FS2D =    0
   6)  -  FBE +  FEB -  FCE +  FEC +  FES1 +  FES3 -  FS1E =    0
   7)  -  FAS1 +  FS1A -  FDS1 +  FS1D -  FES1 +  FS1E -  FS4S1 =    0
   8)  -  FBS2 -  FDS2 +  FS2D +  FS2B -  FS4S2 =    0
   9)     F -  FCS3 -  FDS3 -  FES3 =    0
  10)  -  F +  FS4S1 +  FS4S2 =    0
  11)     FAS1 <=   25
  12)     FS1A <=   25
  13)     FAB <=   6
  14)     FBA <=   6
  15)     FAC <=   3
  16)     FCA <=   3
  17)     FAD <=   4
  18)     FDA <=   4
  19)     FBC <=   2
  20)     FCB <=   2
  21)     FBD <=   7
  22)     FDB <=   7
  23)     FBE <=   8
  24)     FEB <=   8
  25)     FBS2 <=   25
  26)     FS2B <=   25
  27)     FCD <=   5
  28)     FDC <=   5
  29)     FCE <=   6
  30)     FEC <=   6
  31)     FDS1 <=   1
  32)     FS1D <=   1
  33)     FDS2 <=   4
  34)     FS2D <=   4
  35)     FES1 <=   3
  36)     FS1E <=   3
  37)     FCS3 <=   25
  38)     FDS3 <=   6
  39)     FES3 <=   4
```

```
 40)     FS4S2 <=   1000
 41)     FS4S1 <=   1000
END

:*GO

    LP OPTIMUM FOUND  AT STEP    18

         OBJECTIVE FUNCTION VALUE

 1)         26.00000
```

VARIABLE	VALUE	REDUCED COST
F	26.000000	0.000000
FAB	0.000000	0.000000
FAC	3.000000	0.000000
FAD	4.000000	0.000000
FAS1	0.000000	0.000000
FBA	0.000000	0.000000
FCA	0.000000	1.000000
FDA	0.000000	0.000000
FS1A	7.000000	0.000000
FBC	2.000000	0.000000
FBD	2.000000	0.000000
FBE	7.000000	0.000000
FBS2	0.000000	0.000000
FCB	0.000000	1.000000
FDB	0.000000	0.000000
FEB	0.000000	0.000000
FCD	0.000000	1.000000
FCE	0.000000	1.000000
FCS3	16.000000	0.000000
FDC	5.000000	0.000000
FEC	6.000000	0.000000
FDS1	0.000000	0.000000
FDS2	0.000000	0.000000
FDS3	6.000000	0.000000
FS1D	1.000000	0.000000
FS2D	4.000000	0.000000
FES1	0.000000	0.000000
FES3	4.000000	0.000000
FS1E	3.000000	0.000000
FS4S1	11.000000	0.000000
FS2B	11.000000	0.000000
FS4S2	15.000000	0.000000

ROW	SLACK	DUAL PRICES
2)	0.000000	0.000000
3)	0.000000	0.000000
4)	0.000000	1.000000
5)	0.000000	0.000000
6)	0.000000	0.000000
7)	0.000000	0.000000

8)	0.000000	0.000000
9)	0.000000	1.000000
10)	0.000000	0.000000
11)	25.000000	0.000000
12)	18.000000	0.000000
13)	6.000000	0.000000
14)	6.000000	0.000000
15)	0.000000	1.000000
16)	3.000000	0.000000
17)	0.000000	0.000000
18)	4.000000	0.000000
19)	0.000000	1.000000
20)	2.000000	0.000000
21)	5.000000	0.000000
22)	7.000000	0.000000
23)	1.000000	0.000000
24)	8.000000	0.000000
25)	25.000000	0.000000
26)	14.000000	0.000000
27)	5.000000	0.000000
28)	0.000000	1.000000
29)	6.000000	0.000000
30)	0.000000	1.000000
31)	1.000000	0.000000
32)	0.000000	0.000000
33)	4.000000	0.000000
34)	0.000000	0.000000
35)	3.000000	0.000000
36)	0.000000	0.000000
37)	9.000000	0.000000
38)	0.000000	1.000000
39)	0.000000	1.000000
40)	985.000000	0.000000
41)	989.000000	0.000000

NO. ITERATIONS= 18

DO RANGE(SENSITIVITY) ANALYSIS?
>*NO

:*QUIT

END

DATE
FILMED

6 - 83

DTIC