GEORGIA INSTITUTE OF TECHNOLOGY
OFFICE OF CONTRACT ADMINISTRATION
SPONSORED PROJECT INITIATION

Date: January 17, 1980

Project Title: A Decision Support System for the Outfit Planning Problem

Gr cd

Project No: E-24-609

Project Director: Dr. Leon F. McGinnis, Jr.

Sponsor: U. S. Department of Commerce; Washington, D. C. 20230

Agreement Period: From 9/29/79 Until ~~9/28/80~~ ~~12/31/80~~ 8/31/81 (contract period)

Type Agreement: Contract No. MA-79-SAC-00067

Amount: $49,236 DOC
6,478 GIT (E-24-339)
$55,714 Total

Reports Required: Monthly Progress Reports; Interim Reports; Final Technical Report

Sponsor Contact Person (s):

Technical Matters
Mr. Frederick Seibold (COTR)
Office of Maritime Technology, Room 4622
Maritime Administration
Washington, D. C. 20230
(202) 377-5448

(See Mod. No. 1 for alternate COTR)

Contractual Matters
(thru OCA)
Mr. D. J. Pelmotor, Contracting Officer
U. S. Department of Commerce
OP & ADPM, Room 6527
Contract Administration & Support Divisio
Washington, D. C. 20230
(202) 377-4185

Defense Priority Rating: None

Assigned to: Industrial & Systems Engineering (School/Laboratory)

COPIES TO:

- Project Director
- Division Chief (EES)
- School/Laboratory Director
- Dean/Director—EES
- Accounting Office
- Procurement Office
- Security Coordinator (OCA)
- Reports Coordinator (OCA)
- OCA Research Property Coordinator
- Library, Technical Reports Section
- EES Information Office
- EES Reports & Procedures
- Project File (OCA)
- Project Code (GTRI)
- Other

GEORGIA INSTITUTE OF TECHNOLOGY OFFICE OF CONTRACT ADMINISTRATION

SPONSORED PROJECT TERMINATION SHEET

[illegible]

Date 9/28/81

Project Title: A Decision Support System for the Outfit Planning Problem

Project No: E-24-609

Project Director: Dr. Leon F. McGinnis, Jr.

Sponsor: U. S. Dept. of Commerce; Washington, D.C.

Effective Termination Date: 8/31/81

Clearance of Accounting Charges: 8/31/81

Grant/Contract Closeout Actions Remaining:

- [x] Final Invoice and Closing Documents
- [ ] Final Fiscal Report
- [x] Final Report of Inventions
- [x] Govt. Property Inventory & Related Certificate
- [ ] Classified Material Certificate
- [ ] Other ______

Assigned to: Industrial & Systems Engineering (School/Laboratory)

COPIES TO:

Administrative Coordinator	Research Security Services	EES Public Relations (2)
Research Property Management	~~Reports Coordinator (OCA)~~	Computer Input
Accounting	Legal Services (OCA)	Project File
Procurement/EES Supply Services	Library	Other ______

FORM OCA 10:781

E-24-609

GEORGIA INSTITUTE OF TECHNOLOGY

SCHOOL OF INDUSTRIAL AND SYSTEMS ENGINEERING

Atlanta, Georgia 30332

(404) 894-2300

February 13, 1980

Mr. Bob Schaffran
Advanced Ship Development Office
U.S. Department of Commerce
Maritime Administration (M-940)
Washington, D.C. 20230

RE: Contract No. MA79SAC00067
Ga. Tech Project No. E-24-609

MONTHLY REPORT FOR JANUARY, 1980

Dear Mr. Schaffran:

We have begun work on the above referenced contract. During the month of January, several important events occurred. First, Dr. Graves and I met in Amherst to plan our project activities and to initiate some of them. Second, we have received from NAASCO an offer to provide production information on a Carlesbad-class tanker. At this point, we are quite interested, but there is a possibility that the data will not be ready in time for our testing.

I anticipate no problems with our work at this time. If there is anything further I might provide, please let me know.

Sincerely,

[illegible]

Leon F. McGinnis, Jr.
Assistant Professor

LFM:vld

cc: Mr. J. Garvey, MarAd
Mr. F. Siebold, MarAd
Dr. R. Graves, U. Mass.
Dwayne Hutchison ~~Ms. N. McHan,~~ Ga. Tech
File

[illegible]

E-24-609

SCHOOL OF INDUSTRIAL AND SYSTEMS ENGINEERING

Atlanta, Georgia 30332 (404) 894-2300

March 10, 1980

Mr. Bob Schaffran
Advanced Ship Development Office
U.S. Department of Commerce
Maritime Administration (M-940)
Washington, D.C. 20230

RE: Contract No. MA79SAC00067
Ga. Tech Project No. E-24-609

MONTHLY REPORT FOR FEBRUARY, 1980

Dear Mr. Schaffran:

Work is continuing on the above referenced contract. At Georgia Tech, we are beginning to develop the experimental software that will be used in the solution procedures. At U. Mass., Dr. Graves is continuing his development of the evaluation procedure. No problems are anticipated at this time.

If there is anything further I might provide, please let me know.

Sincerely,

[illegible]

Leon F. McGinnis, Jr.
Assistant Professor

LFM:vld

cc: Mr. J. Garvey, MarAd
Mr. F. Siebold, MarAd
Dr. R. Graves, U. Mass.
Mr. Duane Hutchison, Ga. Tech ✓
File

input sent to [illegible]
[illegible]

C-24-609

SCHOOL OF INDUSTRIAL AND SYSTEMS ENGINEERING

Atlanta, Georgia 30332 (404) 894-2300

May 15, 1980

Mr. Bob Schaffran
Advanced Ship Development Office
U.S. Department of Commerce
Maritime Administration (M-940)
Washington, D.C. 20230

RE: Contract No. MA79SAC00067
Ga. Tech Project No. E-24-609

MONTHLY REPORT FOR APRIL, 1980

Dear Mr. Schaffran:

Work is continuing on the above referenced contract. Several alternative solution strategies are currently being evaluated. We will soon be selecting one and starting its detailed implementation. At this time, the project is proceeding as planned.

If there is anything further I might provide, please let me know.

Sincerely,

[illegible]

Leon F. McGinnis, Jr.
Assistant Professor

LFM:vld

cc: Mr. J. Garvey, MarAd
Mr. F. Siebold, MarAd
Dr. R. Graves, U. Mass.
Mr. Duane Hutchison, Ga. Tech
File

E-24-609

SCHOOL OF INDUSTRIAL AND SYSTEMS ENGINEERING

Atlanta, Georgia 30332 (404) 894-2300

June 10, 1980

Mr. Bob Schaffran
Advanced Ship Development Office
U.S. Department of Commerce
Maritime Administration (M-940)
Washington, D.C. 20230

RE: Contract No. MA79SAC00067
Ga. Tech Project No. E-24-609

MONTHLY REPORT FOR MAY, 1980

Dear Mr. Schaffran:

Work is continuing on the above referenced contract. We have selected the most promising of several solution strategies and are proceeding with the detailed software implementation. At this time, the project is proceeding as planned.

If there is anything further I might provide, please let me know.

Sincerely,

[illegible]

.. McGinnis, Jr.
Assistant Professor

LFM:pcp

cc: Mr. J. Garvey, MarAd
Mr. F. Siebold, MarAd
Dr. R. Graves, U. Mass.
Mr. Duane Hutchison, Ga. Tech ✓
File

E-24-609

SCHOOL OF INDUSTRIAL AND SYSTEMS ENGINEERING

Atlanta, Georgia 30332 (404) 894-2300

July 10, 1980

Mr. Bob Schaffran
Advanced Ship Development Office
U.S. Department of Commerce
Maritime Administration (M-940)
Washington, D.C. 20230

RE: Contract No. MA79SAC00067
Ga. Tech Project No. E-24-609

MONTHLY REPORT FOR JUNE, 1980

Dear Mr. Schaffran:

Work is continuing on the above referenced contract. Development of prototype software is proceeding as planned. Dr. Graves and I met in Atlanta for several days this month to finalize our work plans for the remainder of the summer. At this time, the project is proceeding on schedule.

If there is anything further I might provide, please let me know.

Sincerely,

[illegible]

Leon F. McGinnis, Jr.
Associate Professor

LFM:vld

cc: Mr. J. Garvey, MarAd
Mr. F. Siebold, MarAd
Dr. R. Graves, U. Mass.
Mr. Duane Hutchison, Ga. Tech ✓
File

E-24-609

SCHOOL OF INDUSTRIAL AND SYSTEMS ENGINEERING

Atlanta, Georgia 30332 (404) 894-2300

August 13, 1980

Mr. Bob Schaffran
Advanced Ship Development Office
U.S. Department of Commerce
Maritime Administration (M-940)
Washington, D.C. 20230

RE: Contract No. MA79SAC00067
Ga. Tech Project No. E-24-609

MONTHLY REPORT FOR JULY, 1980

Dear Mr. Schaffran:

Work is continuing on the above referenced contract. At Tech, we are still working on software development, and this work is progressing satisfactorily. Dr. Graves, at U. Mass., has been working with Fred Petersen (NAASCO) to obtain some actual outfit activity descriptions for a test problem. I have contacted Dick Wise at IITRI and hope to attend the REAPS Symposium in October. Perhaps we will be able to convene our Industry Advisory Panel at that time.

Currently, the project is proceeding on schedule. If there is anything else I might provide, please let me know.

Sincerely,

Leon F. McGinnis, Jr.
Associate Professor

LFM:vld

cc: Mr. J. Garvey, MarAd
Mr. F. Siebold, MarAd
Dr. R. Graves, U. Mass.
Mr. Duane Hutchison, Ga. Tech ✓
File

Logged 1/19/81
DH

E-24-609

SCHOOL OF INDUSTRIAL AND SYSTEMS ENGINEERING

Atlanta, Georgia 30332

(404) 894-2300

September 5, 1980

Mr. Bob Schaffran
Advanced Ship Development Office
U.S. Department of Commerce
Maritime Administration (M-940)
Washington, D.C. 20230

RE: Contract No. MA79SAC00067
Ga. Tech Project No. E-24-609

MONTHLY REPORT FOR AUGUST, 1980

Dear Mr. Schaffran:

Work is continuing on the above referenced contract. The software development effort has progressed as planned, and some very promising progress has been made in test problem generation. Dr. Graves and I have made plans to attend the REAPS Symposium and to meet with Lou Chirillo, John Mason and John Lightbody.

It is my intention to ask our Office of Contract Administration to seek a no cost extension for the contract. There are two reasons for seeking an extension. The first is that NAASCO is now being very cooperative in providing detailed information from their network planning programs, including budgets and performance for man-hours by activity. By extending the contract termination date, we will be able to capture a more complete, actual problem, and still have time to create the necessary data base and exercise our analytic procedures. This would not be possible with the current December termination date. Moreover, the availability of the data has been out of our control, and, in fact, we had originally planned to use a completely fictional test problem.

The second reason for seeking the extension is somewhat more mundane but nevertheless important. The graduate student who was working on the project here has left school to return to the west coast. Since he had been working on the project for a year, he has some very specialized knowledge and skills related to algorithm and software development required in the project. It will not be possible to replace him immediately, which means that I will have to assume his work as well as my own. This creates a problem, since my time is already committed for the Fall quarter.

If you like I can provide you with a revised schedule, based on extending the project through June 1, 1981. If there is anything else I might provide, please let me know.

Sincerely,

[illegible]

Leon F. McGinnis, Jr.
Associate Professor

LFM:vld

cc: Mr. J. Garvey, MarAd
Mr. F. Seibold, MarAd
Dr. R. Graves, U. Mass.
Mr. Duane Hutchison, Ga. Tech ✓
File

SCHOOL OF INDUSTRIAL AND SYSTEMS ENGINEERING

Atlanta, Georgia 30332 (404) 894-2300

November 6, 1980

Mr. Bob Schaffran
Advanced Ship Development Office
U.S. Department of Commerce
Maritime Administration (M-940)
Washington, D.C. 20230

RE: Contract No. MA79SAC00067
Ga. Tech Project No. E-24-609

MONTHLY REPORT FOR OCTOBER, 1980

Dear Mr. Schaffran:

Work is continuing on the above referenced contract. During October, Dr. Graves and I attended the REAPS Symposium in Philadelphia. Our presentation was well received and we had many encouraging comments. In addition, we have submitted for publication an abbreviated version of the first year's final report. A copy of the manuscript has already been forwarded to Mr. Siebold.

If there is anything else I might provide, please let me know.

Sincerely,

[illegible]

Leon F. McGinnis, Jr.
Associate Professor

LFM:pcp

cc: Mr. J. Garvey, MarAd
Mr. F. Seibold, MarAd
Dr. R. Graves, U. Mass.
Mr. Duane Hutchison, Ga. Tech
File

SCHOOL OF INDUSTRIAL AND SYSTEMS ENGINEERING

Atlanta, Georgia 30332 (404) 894-2300

March 12, 1981

Mr. Bob Schaffran
Advanced Ship Development Office
U.S. Department of Commerce
Maritime Administration (M-940)
Washington, D.C. 20230

RE: Contract No. MA79SAC00067
Ga. Tech Project No. E-24-609

MONTHLY REPORT FOR FEBRUARY, 1981

Dear Mr. Schaffran:

Approval has been granted for the requested no cost extension, with the new termination date 6/1/81. Dr. Graves is nearing completion of a test problem derived from data provided by NAASCO. My work has been directed toward the specification and implementation of computer codes for solving the problem as proposed in our first year report.

If there is anything else I might provide, please let me know.

Sincerely,

Leon F. McGinnis, Jr.
Associate Professor

LFM:vld

cc: Mr. J. Garvey, MarAd
Mr. F. Seibold, MarAd
Dr. R. Graves, U. Mass
Mr. Duane Hutchison, Ga. Tech ✓
File

Cy to OHR

E-24-609

SCHOOL OF INDUSTRIAL AND SYSTEMS ENGINEERING

Atlanta, Georgia 30332 (404) 894-2300

May 13, 1981

Mr. Bob Schaffran
Advanced Ship Development Office
U.S. Department of Commerce
Maritime Administration (M-940)
Washington, D.C. 20230

Re: Contract No. MA79SAC00067
Ga. Tech Project No. E-24-609

MONTHLY REPORT FOR APRIL 1981

Dear Mr. Schaffran:

Work is continuing on the above referenced contract. Dr. Graves has completed the development of a test problem generation scheme, and is now preparing a set of problems to be sent to me on magnetic tape. Algorithmic development is virtually complete and there do not appear to be any significant problems with completing the software.

As noted in the report for last month, I am directing our contracting office to request a no cost extension of the project termination to August 31. This time is to allow for preparation and publication of the final report, and is consistent with our original proposal. No personal services fund for research will be expended during this period.

If there is anything else I might provide, please let me know.

Sincerely,

Leon F. McGinnis
Associate Professor

LFM:vld

cc: Mr. J. Garvey, MarAd
Mr. F. Seibold, MarAd
Dr. R. Graves, U. Mass
Mr. Duane Hutchison, Ga. Tech ✓
File

XC: OHR

E-24-609

SCHOOL OF INDUSTRIAL AND SYSTEMS ENGINEERING

Atlanta, Georgia 30332

(404) 894-2300

April 10, 1981

Mr. Bob Schaffran
Advanced Ship Development Office
U.S. Department of Commerce
Maritime Administration (M-940)
Washington, D.C. 20230

RE: Contract No. MA79SAC00067
Ga. Tech Project No. E-24-609 I54E

MONTHLY REPORT FOR MARCH 1981

Dear Mr. Schaffran:

Work is continuing on the above referenced contract. I expect to have the test problem from Dr. Graves by the end of April.

I note that the schedule of deliverables calls for the final technical report at the end of the contract period, June 1, rather than 90 days later, as proposed. I intend to discuss this with our local contracting officer, and if necessary will request a contract modification to have this delivery date conform to the proposal.

If there is anything else I might provide, please let me know.

Sincerely,

[illegible]

Leon F. McGinnis
Associate Professor

LFM:pcp

cc: Mr. J. Garvey, MarAd
Mr. F. Seibold, MarAd
Dr. R. Graves, U. Mass.
Mr. Duane Hutchison, GA Tech ✓ forward to PPC DH
File

E-24-609
ISyE

SCHOOL OF INDUSTRIAL AND SYSTEMS ENGINEERING

Atlanta, Georgia 30332 (404) 894-2300

July 7, 1981

Mr. Bob Schaffran
Advanced Ship Development Office
U.S. Department of Commerce
Maritime Administration (M-940)
Washington, D.C. 20230

Re: Contract No. MA79SAC00067
Ga. Tech Project No. E-24-609

MONTHLY REPORT FOR MAY AND JUNE, 1981

Dear Mr. Schaffran:

Work is continuing on the above referenced contract. We have received approval to extend the project termination to August 31, 1981. During the month of June, I attended the SP-8 meeting in Portsmouth, NH, and had productive conversations with John Mason and Rodney Robinson, among others.

If there is anything else I might provide, please let me know.

Sincerely,

Leon F. McGinnis
Associate Professor

LFM:vld

cc: Mr. J. Garvey, MarAd
Mr. F. Seibold, MarAd
Dr. R. Graves, U. Mass
Mr. Duane Hutchison, Ga. Tech ✓
File

- -r-609

GEORGIA INSTITUTE OF TECHNOLOGY

SCHOOL OF INDUSTRIAL AND SYSTEMS ENGINEERING

Atlanta, Georgia 30332 (404) 894-2300

August 18, 1981

Mr. Bob Schaffran
Advanced Ship Development Office
U.S. Department of Commerce
Maritime Administration (M-940)
Washington, D.C. 20230

Re: Contract No. MA79SAC00067
Ga. Tech Project No. E-24-609

MONTHLY REPORT FOR JULY, 1981

Dear Mr. Schaffran:

Work is continuing on the above referenced contract. Barring unforeseen delays, the final report should be completed by August 31 and on your desk soon after.

If there is anything else I might provide, please let me know.

Sincerely,

[illegible]

Leon F. McGinnis
Associate Professor

LFM:vld

cc: Mr. J. Garvey, MarAd
Mr. F. Seibold, MarAd
Dr. R. Graves, U. Mass.
Mr. Duane Hutchison, GA Tech
File

XC: Mr. O.H. Rodgers

FINAL TECHNICAL REPORT

A DECISION SUPPORT SYSTEM FOR THE OUTFIT PLANNING PROBLEM: ANALYSIS AND SOLUTION METHODOLOGY

By

Dr. Leon F. McGinnis
Georgia Institute of Technology

and

Dr. Robert J. Graves
University of Massachusetts

Prepared for
DEPARTMENT OF COMMERCE

Under
Contract No. MA79SAC00067

August 31, 1981

GEORGIA INSTITUTE OF TECHNOLOGY
A UNIT OF THE UNIVERSITY SYSTEM OF GEORGIA
SCHOOL OF INDUSTRIAL & SYSTEMS ENGINEERING
ATLANTA, GEORGIA 30332

1981

REPORT DOCUMENTATION PAGE	1. REPORT NO.	2.	3. Recipient's Accession No.
4. Title and Subtitle A Decision Support System for the Outfit Planning Problem: Analysis and Solution Methodology			5. Report Date August 31, 1981
			6.
7. Author(s) Leon F. McGinnis, Jr., and Robert J. Graves (University of Mass. - Amherst)			8. Performing Organization Rept. No.
9. Performing Organization Name and Address Georgia Tech Research Institute Administration Building Georgia Institute of Technology Atlanta, Georgia 30332			10. Project/Task/Work Unit No.
			11. Contract(C) or Grant(G) No. (C) MA79SAC00067 (G)
12. Sponsoring Organization Name and Address U.S. Department of Commerce Maritime Administration 14th St. and Constitution Ave., N.W. Washington, D.C. 20230			13. Type of Report & Period Covered Final (9/79-8/81)
			14.

15. Supplementary Notes

16. Abstract (Limit: 200 words)

In the on-unit/on-block/on-board, or zone approach to outfitting, a fundamental problem is to select the set of outfitting activities to be performed on-block. The primary constraints limiting on-block outfitting are the time available and outfitting labor. The selected activities must be performed within a given outfitting window and labor availability profile.

This complex resource allocation problem has been modelled as an optimization problem. This report presents a methodology for analyzing the problem, based on the optimization model. In a two-phase approach, a set of activities is first selected to maximize the benefit for on-block outfitting subject to time available and total labor available. In the second phase, a resource feasible schedule is constructed.

The selection and scheduling problems both require new methods for their solution. Algorithm development is presented, along with empirical evaluation based on a set of randomly generated test problems. The methodology appears to be quite suitable for solving the outfit planning problem.

17. Document Analysis a. Descriptors

Shipbuilding, Industrial Engineering, Project Scheduling, Resource Allocation, Network Modelling, Heuristic Optimization

b. Identifiers/Open-Ended Terms

c. COSATI Field/Group

18. Availability Statement	19. Security Class (This Report)	21. No. of Pages
Release Unlimited	Unclassified	130
	20. Security Class (This Page) Unclassified	22. Price

(See ANSI-Z39.18) See Instructions on Reverse OPTIONAL FORM 272 (4-
(Formerly NTIS-35)

A DECISION SUPPORT SYSTEM FOR THE OUTFIT PLANNING PROBLEM: ANALYSIS AND SOLUTION METHODOLOGY

August 31, 1981

Final Technical Report

Department of Commerce Contract

No. MA79SAC00067

ABSTRACT

In the on-unit/on-block/on-board, or zone approach to outfitting, a fundamental problem is to select the set of outfitting activities to be performed on-block. The primary constraints limiting on-block outfitting are the time available and outfitting labor. The selected activities must be performed within a given outfitting window and labor availability profile.

This complex resource allocation problem has been modelled as an optimization problem. This report presents a methodology for analyzing the problem, based on the optimization model. In a two-phase approach, a set of activities is first selected to maximize the benefit for on-block outfitting subject to time available and total labor available. In the second phase, a resource feasible schedule is constructed.

The selection and scheduling problems both require new methods for their solution. Algorithm development is presented, along with empirical evaluation based on a set of randomly generated test problems. The methodology appears to be quite suitable for solving the outfit planning problem.

ACKNOWLEDGEMENTS

This report is the culmination of a research effort that has spanned four years and two universities. A number of individuals in the universities, in MarAd and in the shipbuilding industry have contributed to the direction and execution of the research. We would specifically like to acknowledge the contributions of our research assistants, Linda Bailey, Al Beattie and Kahan Talebi. In addition, the following members of our Industry Advisory Panel have generously contributed time, advice, critique and encouragement:

Lou Chirillo
Research and Development Program Manager
Todd-Pacific Shipyards Corporation

John Mason
MarAd Program Manager
Bath Iron Works Corporation

Doug Peel
Outfitting Superintendent
Sun Shipbuilding and Drydock Company

Fred Peterson
National Steel and Shipbuilding Company

Robert Schaffran and John Garvey, of MarAd, have been very helpful and assisted in establishing industry dialogue. Finally, Mrs. Vicki DeLoach has done a masterful job on the correspondence and report preparation.

TABLE OF CONTENTS

production resources involved in hull and outfitting work. For the planning problem, in particular, there are no well established practices or guidelines to assist in the process of integrating hull and outfit work. A first step toward solving this problem is the work reported in [27], where the <u>outfit planning problem</u> is stated as follows:

Given:
(1) a catalog of the outfit phase activities for which there are outfitting options;

(2) for each such activity, a list of the outfitting options, including time, resource and precedence requirements;

(3) the ship delivery schedule and any fixed milestone deadlines;

(4) labor availability by craft and grade;

(5) facility capacities and availabilities (lifting, covered space, yard space, etc.); and

(6) other constraining factors (material availability, rate of cost accumulation, etc.).

Determine: The outfitting option to be used for each outfit phase activity considered, along with the necessary schedule.

The development of the conceptual and mathematical model in [27] is based on the three outfitting stages, on-unit, on-block, and on-board, as presented in [9]. Building upon this, the outfit components are classified in [27] as follows:

(1) on-board components - furnishings, equipment or other materials which are subject to damage or pilferage and are always installed in the on-board stage;

(2) non-unit components - isolated components or components in distributed systems, e.g., wireways, ventilation ducting, which are not candidates for on-unit outfitting;

(3) free components - components which may either form a unit or may be outfitted separately.

Non-unit components may be outfitted on-block or on-board, and free components can be outfitted in any stage. If on-unit outfitting is selected, the resulting unit may be installed either on-block or on-board.

In [27] the definition of potential units was given in terms of a minimum outfit kit and a maximum outfit kit. Activities (work packages) were assumed for each possible outfitting option for each outfit component or kit. The outfit planning problem then becomes one of selecting the best feasible set of activities (or work packages). Further details of the formulation can be found in [27].

The formulation in [27], while a good beginning, exhibits some conceptual and pragmatic shortcomings. In the first place, it doesn't incorporate a consideration of the operating organization within which outfitting is actually performed. More serious, however, is the assumption that a great many options will be evaluated in detail prior to the planning process.

The work reported here uses the models in [27] as a jumping off point. That formulation is modified in light of practical considerations, and is placed in a realistic setting. Based on this problem setting, analytic procedures are developed for making outfit planning decisions, i.e., deciding which outfit components or units to install on-block, and for guiding the allocation of outfit labor to blocks.

The remainder of this chapter contains the discussion of the problem setting, the solution approach and some relevant implementation issues. Chapter 2 presents a methodology for selecting components or units for on-block outfitting and Chapter 3 presents a methodology for determining a best feasible schedule for the selected activities. The computer implementation of these methodologies is discussed in Chapter 4 and the generation of test problems is discussed in Chapter 5. Computational results are presented and analyzed in Chapter 6 and Chapter 7 summarizes the work and prints out needed additional research.

1.1 PROBLEM SETTING

The outfit planning problem arises at that point in the planning process where the block erection schedules and labor allocation decisions are being made. Thus, work packages may only contain a description of the associated components. Detailed process plans or work instructions may not be available. However, it is assumed that:

A1: there are standard hours factors that can be used to estimate the labor hours associated with each component and unit for each possible outfit stage; and

A2: a reasonable estimate of the appropriate duration for a work package can be obtained.

The standards may correspond to historical performance data or to engineered standards (see for example, [1], describing current developments).

Typically, an outfitting activity may require several different crafts. At this point in the planning process, however, it will usually be difficult to obtain an estimate other than the total labor required. Thus, the labor hours can be viewed as a weighted average of all crafts involved. For planning purposes, this appears to be a reasonable assumption, especially if the mix of craft labor is reasonably uniform across outfitting work packages within a block.

Even if the mix varies considerably among the work packages, the mix of labor assigned to a block or available to be assigned to a block will remain fairly constant. Any inaccuracies in this assumption at the planning stage can be accommodated by the flexibility available to production supervision at the work site. In contrast to the formulation in [27], it will therefore be assumed that

A3: there is a single category of labor required, referred to simply as "outfitting labor."

This assumption simplifies the outfit planning problem somewhat, without compromising its realism.

The benefits of on-unit outfitting are now well recognized [9]. Moreover, an informal survey of shipyards indicates that unit fabrication is not a bottleneck in production. Thus, any potential unit which is technically and economically feasible should <u>always</u> be produced. The only decision is whether to install the unit on-block or on-board.

Further, it seems quite unlikely that the design process (at least in the foreseeable future) will be flexible enough to define minimum and maximum outfit kits. In practice, the most realistic expectation would be for a fixed unit definition with regard to components. Thus, departing from the model in [27], it is assumed that:

A4: a catalog of fixed unit designs is given for on-unit outfitting; there is still the question of whether to install the unit on-block or on-board.

Not only does this assumption reduce the complexity of the model and the magnitude of the data required, it also conforms more closely with the realities of ship design and production engineering.

At this point, there is a catalog of components and units which may be installed either on-block or on-board, and for each one, there is an estimate of the total labor hours and duration for either outfitting stage. Since it is presumed that on-block outfitting is preferred, the ideal solution would call for on-block outfitting of all available units and components. This ideal solution will not generally be feasible.

At least two factors limit the amount of on-block outfitting possible for any block. The first is the "outfitting window," or the time available for outfitting. Outfitting on-block cannot begin until there is a sufficient hull structure to support outfitting, and must cease prior to transporting the block to the hull erection site. Especially if there are technological precedence requirements among outfit work packages, this window may not be long enough to encompass all possible activities.

The second constraining factor is the amount of outfitting labor available for a particular block and the timing of availability. Because of space limitations and congestion considerations, there is an upper limit on the size of the outfitting crew assigned to a block at one time. In addition, because of start up and stop work effects, the maximum crew size is diminished at the start and the end of the outfitting window. This leads to the following assumption:

A5: outfitting labor is allocated to blocks according to the pattern illustrated in Figure 1.1.

The methods reported here do not specifically address the problem of allocation to the blocks, but they do provide valuable information for guiding the allocation process.

Other factors, such as block weight, may constrain the selection of units and components for installation on-block. While these factors are not considered in the methods presented here, these methods could be generalized to incorporate certain additional constraints.

If not all possible components and units can be installed on-block, then the outfit planning problem is to decide which should be selected (or conversely, which should be deferred for on-board outfitting) and to demonstrate a feasible schedule. The criterion, as formulated in [27], is to minimize total outfitting cost, or equivalently, to maximize the savings for on-block outfitting. Thus:

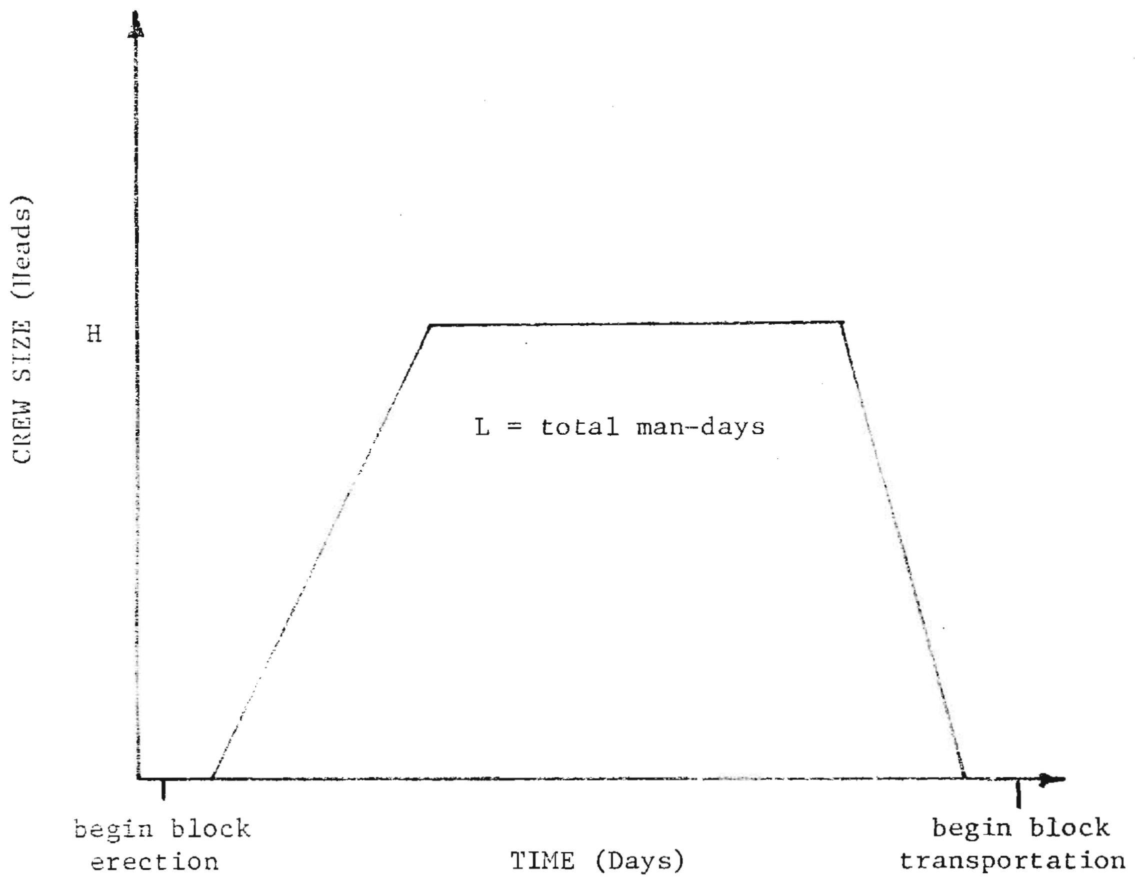

Figure 1.1: Outfit Labor Allocation Profile

A6: for each outfit component and unit there are two cost estimates, corresponding to on-block and on-board installation.

These cost estimates could be obtained, for example, from the labor hour estimates plus average factors for overhead and utilities.

1.2 METHODOLOGICAL APPROACH

In this problem setting, it is natural to decompose the planning problem by blocks. For the outfitting material associated with a particular block, there is a given set of units which will be fabricated and a set of non-unit components; both sets are available for on-block outfitting, but any items may be deferred to the on-board stage.

The methodology developed here for solving the outfit planning problem for a particular block is heuristic in nature and involves two distinct phases. The first phase, activity selection, produces an "optimal" set of activities for on-block outfitting. This set of activities maximizes the potential savings, is feasible with respect to the outfitting window, and with respect to total available outfitting labor.

The activity selection phase does not consider the pattern of outfit labor availability (Figure 1.1). It is possible that there is no way to schedule the selected set of activities within the given pattern. The second phase, activity scheduling, develops a labor feasible schedule, although it may require dropping some activities from the selected set.

The activity selection problem is solved by generalizing the analysis of a similar problem in [44]. The problem solved in [44] did not include a labor resource budget, although it was discussed as an extension of the model. In solving the activity selection problem, a Lagrangian relaxation [20] is used to identify realizable optimal labor allocations as well as the corresponding activity selections. This information is efficiently generated, even for large problems, and could provide valuable guidance for the labor allocation process.

The activity scheduling problem is solved by a new procedure for the resource constrained project scheduling problem [5 , 12 , 15 , 45]. The new and unique feature of this problem is that the available resource profile is not constant, but has periods of increasing resources, constant resources, and decreasing resources. The solution procedure incorporates two extensions of the traditional "SPAR-type" algorithm [41 , 58]: activity durations are allowed to vary within a fixed range, maintaining a constant total labor requirement; and the scheduling rule is modified to accommodate a decreasing resource availability.

The assumption that activity durations may vary within some narrow range is easily defensible. At this point in the planning process, detailed analysis of work packages has not been done, and likewise, it is impossible to do detailed (man-by-man) labor assignment. The flexibility in activity duration merely reflects a lack of certainty about what the actual duration will be or should be. In fact, the solution to the scheduling problem will provide a desirable target duration.

The manner in which the duration is allowed to vary could well result in a fractional crew size. For example, suppose the standard data estimate for labor is 200 hours and the duration is expected to be 5 days. The resulting crew size would be 200 ÷ [5 x 8] = 5 heads. However, if the duration is allowed to be either 4 days or 6 days, the corresponding crew size is 200 ÷ [4 x 8] = 6.25 heads or 200 ÷ [6 x 8] = 4.17 heads. Again, this presents no practical problem, since varying work assignments at the work site will yield the desired duration.

1.3 IMPLEMENTATION ISSUES

The model formulation given in [27] has been modified in several respects. These modifications accomplish two objectives:

(i) they bring the problem formulation more in line with the reality of design, engineering, and production; and

(ii) they render the resulting mathematical model amenable to well structured and computationally efficient heuristics.

It is important to note that, although the computational tests reported here were conducted on a large main frame computer, the methods used are well suited to implementation on minicomputers which support FORTRAN. For example, the programs developed for the two solution procedures could easily be modified to solve realistic problems on a machine such as the HP-1000.

The modified formulation also has much more realistic data requirements. The industry is moving relentlessly toward standard data for production. Ultimately, this should make possible early estimation of labor hours for outfitting work packages. In particular, estimating direct labor plus allowances for outfit stage should give sufficiently accurate values for the outfit planning methods developed here.

Cost estimation also benefits from the development of standard data. Using labor hour estimates as a base, and cost factors for outfit stage (e.g., to include cost of utilities on-board) reasonable cost estimates should be obtainable with reasonable effort.

In practice, the movement is toward computerized standard data systems. The availability of computerized standard data admits the possibility of computer generating much of the data required in the outfit planning model.

In summary, the model and approach are in tune with current practice and developments in shipbuilding. There are few institutional or practical (in the long run) problems to prevent implementation of the methods developed.

2.0 ACTIVITY SELECTION

The goal of activity selection is to choose the set of outfit components and units to be installed on block so that:

(1) the greatest benefit, i.e., cost reduction, is obtained,

(2) the set chosen is feasible with respect to labor available for outfitting, and

(3) considering only precedence constraints, it is possible to complete all selected activities within the block outfitting window.

In the procedure described here, the set of activities selected does give the maximum cost reduction, and does not exceed the <u>total</u> available man-days of outfitting on the block.

It is possible that the set of activities cannot actually be scheduled within the given day by day profile of outfitting labor. However, if there are "enough" activities in the set, and if their durations are "sufficiently" flexible, then, almost surely, a feasible schedule can be found.

The third requirement is satisfied by screening out all activities which, based on shortest durations and CPM early finish times, could never be completed within the available outfitting window. This is referred to in the following as "time screening."

2.1 MATHEMATICAL MODEL

Consider a single hull block. For this hull block, the erection schedule and work packages can be used to define an outfitting "window" or time period of length T during which outfitting may be performed. No outfitting work may begin prior to the window, nor continue beyond the window. Within the window, a total of L man-days of outfitting work can be accomplished.

For this block, there is a catalog of potential outfitting work packages, or activities, A. For activities in A, there are technological restrictions, or precedence requirements, on activity work sequence. Thus each activity, j, is described by a set of parameters:

v_j:	the savings or value for doing this activity on block rather than deferring it to the on-board mode
L_j:	total outfitting labor required for this activity
ℓ_j:	minimum possible duration
u_j:	maximum allowable duration
p(j):	set of activities which must precede activity j
s(j):	set of activities which j must precede.

The activity selection problem is to select a subset of activities in A, such that

(1) if j is selected, then all activities in p(j) are selected;

(2) if j is selected, it can be scheduled to finish before time T;

(3) the total labor required by selected activities does not exceed L; and

(4) the value of the selected activities is maximized.

Note that the selection of T and L are themselves decision problems of considerable importance, since they obviously limit the capacity for on block outfitting.

A mathematical statement of the activity selection problem is

$$(SP_{TL}) \qquad V(T, L) = \text{maximum} \sum_{j \in A} v_j y_j$$

$$\text{subject to} \quad -t_i y_i + t_j y_j \geq d_j \qquad j \in A;\ i \in p(j)$$

$$-y_i + y_j \leq 0 \qquad j \in A;\ i \in p(j)$$

$$\sum_j L_j y_j \le L$$

$$y_j \in \{0, 1\} \qquad j \in A$$

$$0 \le t_j \le T - d_j \qquad j \in A$$

where

d_j = duration chosen for activity j, $d_j \in [\ell_j, u_j]$

t_j = CPM early finish time for activity j

$$y_j = \begin{cases} 1 \text{ if activity } j \text{ is selected} \\ 0 \text{ otherwise} \end{cases}$$

With given values for d_j, this problem is a generalization of the so-called Project Coordinator's Problem treated by McGinnis and Nuttle [44]. While they suggested an approach to solving this generalization of their problem, no detailed development was presented.

As in [44], an initial time screening consists of computing t_j using an appropriate d_j for all activities in A, and dropping from further consideration any activity whose $t_j > T$. After the time screening, the problem reduces to the following pure selection problem.

$$(\text{PSP}_L^T) \qquad V_T(L) = \text{maximum} \sum_{j \in A} v_j y_j$$

$$\text{subject to} \quad -y_i + y_j \le 0 \qquad j \in A,\ i \in p(j) \qquad [w_{ij}]$$

$$\sum_j L_j y_j \le L \qquad [\lambda]$$

$$y_j \in \{0, 1\} \qquad j \in A \qquad [w_j]$$

2.2 SOLVING THE PURE SELECTION PROBLEM

As discussed in [44] this is an integer programming problem, and, in general requires special discrete methods for its solution. The approach for solving this problem suggested in [44] is to exploit the following Lagrangian relaxation [20]:

$$(\mathrm{PSP}^T_{L\lambda}) \qquad V_T(L, \lambda) = \text{maximum} \sum_j v_j y_j + \lambda[L - \sum_j L_j y_j]$$

$$= \text{maximum} \sum_j (v_j - \lambda L_j) y_j + \lambda L$$

$$\text{subject to } -y_i + y_j \leq 0 \qquad j \in A,\ i \in p(j)$$

$$y_j \in \{0, 1\} \qquad j \in A$$

For a given value of λ, $V_T(L, \lambda)$ can be determined by solving the linear programming dual of $(\mathrm{PSP}^T_{L\lambda})$. The dual problem, as shown in [44], is a network flow problem, so its solutions are naturally integer and can be obtained by efficient algorithms, such as [6, 24, 26, 31].

For any value of λ, $V_T(L, \lambda) \geq V_T(L)$, i.e., the Lagrangian relaxation provides an upper bound on the optimal solution value. If, in addition, the solution to $(\mathrm{PSP}^T_{L\lambda})$ satisfies the resource constraint, it provides a feasible solution, or lower bound on $V_T(L)$. The difference between this feasible solution value and $V_T(L, \lambda)$ is referred to as the "gap" and its magnitude is

$$\lambda \left[L - \sum_j L_j y_j \right]$$

For arbitrarily selected λ, the gap is almost always nonzero. In order to "resolve the gap" some form of bounded enumeration or branch and bound [23, 21, 22] will be required in general.

However, the problem being addressed here involves not only the selection of the activities, but the selection of L as well. This fact can be exploited so that the choices of values for L are restricted and for each such choice, an optimal selection of activities can be made by solving $PSP^T_{L\lambda}$ for an appropriate λ. To prove this assertion, consider $V_T(0, \lambda)$ as a function of λ, which is shown in Figure 2.1. Also, $g(\lambda) = \sum_j L_j y_j$ as a function of λ is shown in Figure 2.2.

First, note that in solving $(PSP^T_{L\lambda})$, the term L in the objective function is a constant, thus has no effect on the optimization. This means that if $V_T(L, \hat{\lambda})$ has been obtained for any value of L, then $V_T(L', \hat{\lambda})$ can immediately be computed for any other value by

$$V_T(L', \tilde{\lambda}) = V_T(L, \hat{\lambda}) + \hat{\lambda}(L' - L)$$

Figure 2.1 also illustrates $V_T(L^a, \lambda)$ and $V_T(L^b, \lambda)$, $L^a > L^b > 0$, where the two functions have been normalized so that $V_T(0, \lambda^a) = V_T(L^a, \lambda^a) = V_T(L^b, \lambda^a)$.

Next, observe that both $V_T(\cdot, \lambda)$ and $g(\lambda)$ are piecewise linear, and their breakpoints coincide. Furthermore, $V_T(\cdot, \lambda)$ is convex, while $g(\lambda)$ is a monotonically decreasing step function. Define $\mathcal{L}$ to be the set of realizable values of $g(\lambda)$. The crucial observation is the following:

> If the selected value of L, say L', is in $\mathcal{L}$, i.e., it corresponds to $g(\lambda)$ for some λ, say $\hat{\lambda}$, then there is a zero gap and $(PSP^T_{L'\hat{\lambda}})$ provides an optimal solution to $(PSP^T_{L'})$.

Thus, if the possible choices for L are limited to the set $\mathcal{L}$, an optimal selection can be guaranteed and can be obtained without resorting to discrete optimization methods.

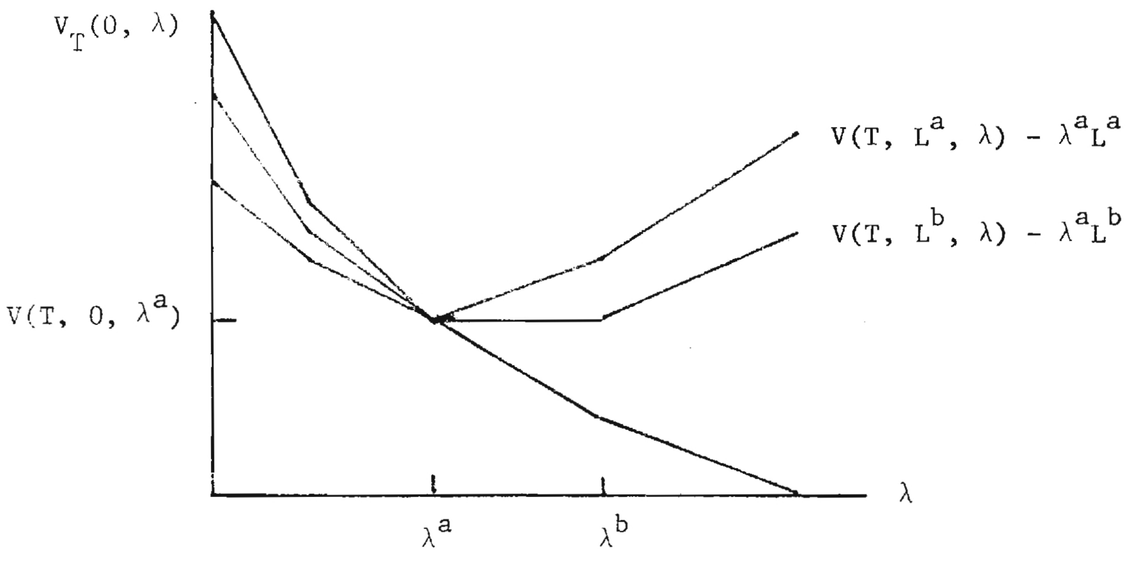

Figure 2.1: Hypothetical $V_T(0, \lambda)$ vs. λ

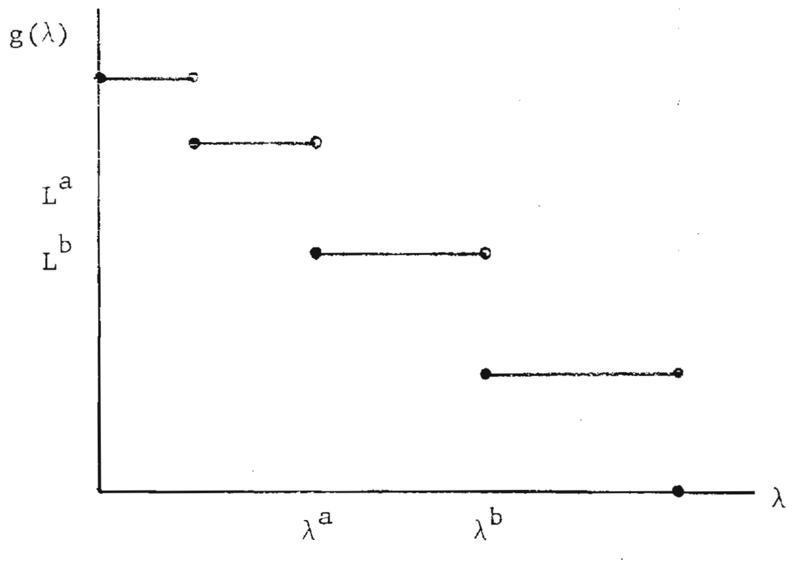

Figure 2.2: Hypothetical $\sum_j L_j y_j$ vs. λ

There are compelling arguments in favor of this approach. In the first place, if the set A is reasonably large, say fifty or more activities, and admits a reasonable degree of parallelism, then the set L should be quite large, with adjacent values close together. The corresponding values of $V_T(\cdot, \lambda)$ should also be close, so that very little savings is given up by the restriction $L \varepsilon L$.

This approach not only provides a means for solving the problem of concern here, the activity selection problem, it also creates information of great value in allocating outfitting labor to blocks, i.e., in determining L for each block. Define λ_k to be the k^{th} breakpoint of $V_T(0, \lambda)$, and let K be the index set for breakpoints. Associated with λ_k is a labor usage $L^k = g(\lambda_k)$ and a value $V_k = V_T(L^k, \lambda_k)$. When allocating labor to each block, the values of (L^k, V_k), $k \varepsilon K$ for each block can guide the process, particularly in making incremental allocation changes.

This approach can easily be extended to allow evaluation of varying windows, or values of T. Suppose the function $V_T(0, \lambda)$ has been generated, by finding all the breakpoints. Now consider $\hat{T} < T$. Clearly $V_{\hat{T}}(0, 0) < V_T(0, 0)$, and the function $V_{\hat{T}}(0, \lambda)$ will be below the function $V_T(0, \lambda)$. By imposing an arbitrary grid on T, the function V(T, L) can be approximated by first specifying T, then generating $V_T(L, \lambda)$ as discussed above.

The following sections of this chapter will detail the development of an algorithm for generating the curve $V_T(0, \lambda)$, and the corresponding sets L and $\{\lambda_k\colon\ k \varepsilon K\}$. Both T and $\{d_j\}$ are parameters in this algorithm. Chapter 4 will describe the computer implementation. The computational evaluation will be presented in Chapter 6.

2.3 ALGORITHM TO GENERATE $V_T(0, \lambda)$

The generation of $V_T(0, \lambda)$ involves two key elements: solving $(PSP^T_{L\lambda})$ for given λ; and locating the breakpoints, λ_k, $k \varepsilon K$. As mentioned earlier, the dual of $(PSP^T_{L\lambda})$ is a minimum cost network flow problem for which several very efficient computer codes are available [6, 24, 26, 31]. Since $V_T(0, \lambda)$ is piecewise linear and convex, the breakpoints can be located by an efficient search procedure. Both these are explained in greater detail below.

2.3.1 The Network Dual Problem

The linear programming dual (relaxing $y_j \varepsilon \{0, 1\}$ to $0 \leq y_j \leq 1$) of $PSP^T_{L\lambda}$ is:

$$(DSP^T_{L\lambda}) \qquad D_T(L, \lambda) = \text{minimum} \sum_j w_j + \lambda L$$

subject to

$$+ \sum_{i \varepsilon p(j)} w_{ij} - \sum_{k \varepsilon s(j)} w_{jk} + w_j - S_j = (v_j - \lambda L_j) \qquad j \varepsilon A$$

$$w_{ij} \geq 0 \qquad w_j \geq 0$$

Complementary slackness conditions require:

(i) $w_j > 0 \Rightarrow y_j = 1$

$w_{ij} > 0 \Rightarrow y_i = y_j$

(ii) $y_j = 1 \Rightarrow S_j = 0$

$S_j > 0 \Rightarrow y_j = 0$

Once the dual solution is obtained, the complementary slackness conditions yield the primal solution trivially.

Figure 2.3 illustrates the network flow model interpretation of $(DST^T_{L\lambda})$.

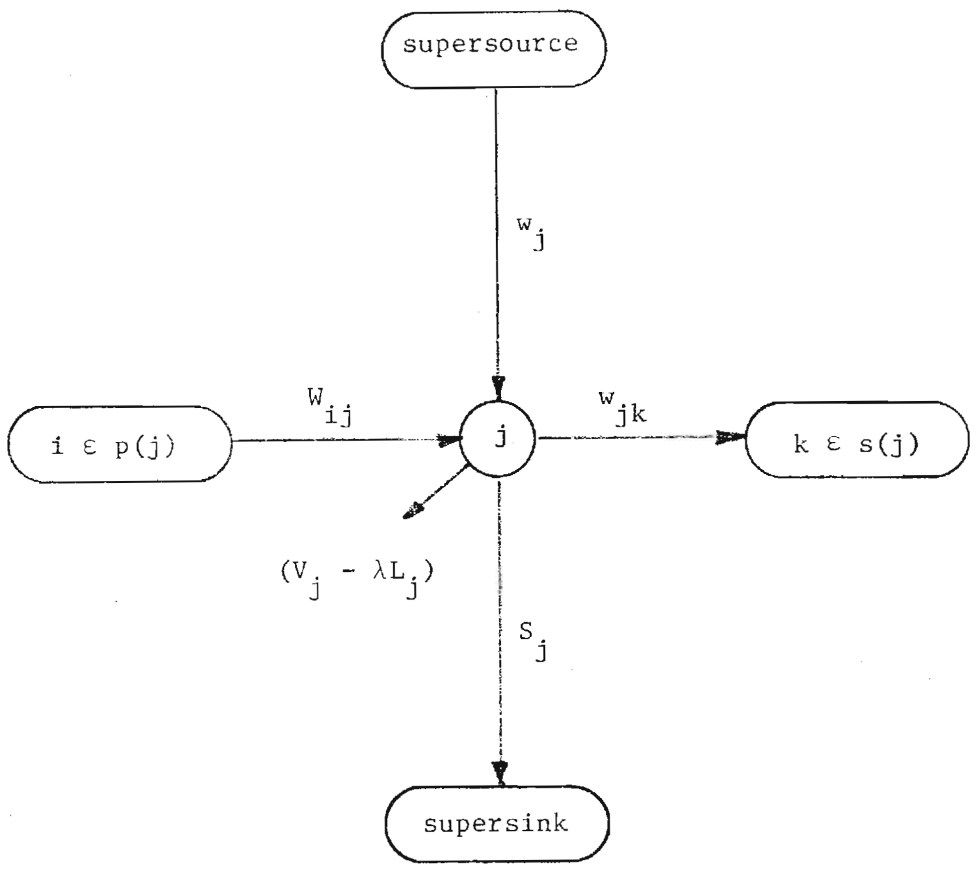

Figure 2.3: Network Interpretation of $(DSP_{T\lambda})$

There is a node in the network for each activity $j \in A$. In Figure 2.3, all arcs incident to a generic node are shown. Note that if $v_j - \lambda L_j < 0$, it represents a required flow into node j, otherwise it is a flow out of node j.

The only technical difficulty in solving $(PSP^T_{L\lambda})$ via the network dual problem is the requirement for integer valued data in the network problem. In general, $v_j - \lambda L_j$ will not be integer. This can be overcome, however, by scaling $(v_j - \lambda L_j)$ by a suitable power of 10, say 10^4, and scaling $D_T(L, \lambda)$ by the inverse, 10^{-4}.

In the search for the breakpoints, $\{\lambda_k: \ k \in K\}$, the network flow problem must be solved many times with slightly different resource vectors $[v_j - \lambda L_j]$. Suppose the problem has just been solved to obtain $D_T(0, \lambda_1)$ and now must be solved for λ_2. The new resource vector is

$$v_j - \lambda_2 L_j = v_j - \lambda_1 L_j - (\lambda_2 + \lambda_1)L_j \qquad j = 1, \ldots, N$$

The vector $[(\lambda_2 + \lambda_1)L_j]$ is a right hand side change vector in linear programming terminology. Thus, to obtain $D_T(0, \lambda_2)$, a parametric analysis of the previous problem may be used, rather than solving a completely new problem.

2.3.2 Searching for Breakpoints

In searching for the breakpoints, several observations about $V_T(0, \lambda)$ are useful. $V_T(0, \lambda)$ is piecewise linear, convex, and decreasing over the range from $\lambda = 0$ to $\lambda = \lambda_{max}$. The largest value of λ which need be considered is that value which just makes all $v_j - \lambda L_j$ nonpositive, i.e.

$$v_j - \lambda_{max} L_j \leq 0 \qquad \forall\, j$$

$$\text{or} \quad \lambda_{max} \geq v_j/L_j \qquad \forall\, j$$

$$\text{or} \quad \lambda_{max} = \max_j \, (v_j/L_j)$$

For any λ which is not a breakpoint, the left hand and right hand derivatives of $V_T(0, \lambda)$ are the same. For λ a breakpoint, the left hand derivative is less than (or equal to) the right hand derivative. The right hand derivative is $-g(\lambda)$. The left hand derivative at a breakpoint, λ, can be determined from $-g(\lambda - \varepsilon)$ where ε is a suitably chosen small positive constant.

Suppose two breakpoints are known, $\lambda_\ell < \lambda_r$, with function values V_ℓ and V_r. If $-g(\lambda_\ell) = -g(\lambda_\ell - \varepsilon)$ then the two breakpoints are adjacent, otherwise there is at least one breakpoint between them. If there is only one breakpoint between them, λ_p, it can be determined immediately by <u>projecting</u> the right hand derivative at λ_ℓ and the left hand derivative at λ_r:

$$\lambda_p = \frac{(V_r - V_\ell) - \lambda_\ell g(\lambda_\ell) + \lambda_r g(\lambda_r - \varepsilon)}{-g(\lambda_\ell) + g(\lambda_r - \varepsilon)}$$

If $-g(\lambda_p) = -g(\lambda_r - \varepsilon)$, then λ_p is adjacent to λ_r, and also adjacent to λ_ℓ, and is the only breakpoint in the interval $(\lambda_\ell, \lambda_r)$. Otherwise, λ_p can be used to split the interval $[\lambda_\ell, \lambda_r]$ into two smaller intervals $[\lambda_\ell, \lambda_p]$, $[\lambda_p, \lambda_r]$ and the process repeated on each of the smaller intervals.

Thus, the search procedure recursively splits an interval into segments until the segments contain a single breakpoint. In order to specify the search algorithm, the following notation is needed:

λ_ℓ:	the left endpoint of an interval
λ_r:	the right endpoint of an interval
L_ℓ, L_r:	$g(\lambda_\ell)$, $g(\lambda_r)$ from the solution of $(PSP^T_{0\lambda_\ell})$, $(PSP^T_{0\lambda_r})$
V_ℓ, V_r:	$D_T(0, \lambda_\ell)$, $D_T(0, \lambda_r)$
S:	set of unresolved intervals
B:	set of breakpoints

An unresolved interval is described by $(\lambda_\ell, \lambda_r, L_\ell, L_r, V_\ell, V_r)$, and a breakpoint is described by (λ_p, L_p, V_p). Note that the endpoints of an unresolved interval need not be breakpoints.

The algorithm specification given below also uses the following short hand descriptions of certain algorithm steps:

SOLVE(λ_p):	solve $DSP^T_{0\lambda_p}$ to determine L_p and V_p
PROJECT(λ_ℓ, λ_r):	compute the projection, λ_p
PUSH(λ_ℓ, λ_r, L_ℓ, L_r, V_ℓ, V_r):	add this interval to S
POP(λ_ℓ, λ_r, L_ℓ, L_r, V_ℓ, V_r):	retrieve an interval from S

The complete specification of the search algorithm can now be given.

```
INITIALIZATION PHASE:

    $\lambda_\ell \leftarrow 0$
    $L_\ell \leftarrow \sum_j L_j$
    $V_\ell \leftarrow \sum_j v_j$
    $\lambda_r \leftarrow \max\{v_j/L_j:\ j = 1, \ldots, n\}$
    $L_r \leftarrow 0$
    $V_r \leftarrow 0$
     $B \leftarrow \{(\lambda_\ell, L_\ell, V_\ell), (\lambda_r, L_r, V_r)\}$
    $\lambda_r \leftarrow \lambda_r - \varepsilon$
    $(L_r, V_r) \leftarrow$ SOLVE($\lambda_r$)
    PUSH($\lambda_\ell, \lambda_r, L_\ell, L_r, V_\ell, V_r$)
```

```
SEARCH PHASE:
  WHILE $S \neq \emptyset$ DO
    POP($\lambda_\ell$, $\lambda_r$, $L_\ell$, $L_r$, $V_\ell$, $V_r$)
    $\lambda_p \leftarrow$ PROJECT($\lambda_\ell$, $\lambda_r$)
    ($L_p$, $V_p$) $\leftarrow$ SOLVE($\lambda_p$)
    WHILE $L_p \neq L_r$ DO
      PUSH($\lambda_p$, $\lambda_r$, $L_p$, $L_r$, $V_p$, $V_r$)
      $\lambda_p \leftarrow$ PROJECT($\lambda_\ell$, $\lambda_p$)
      ($L_p$, $V_p$) $\leftarrow$ SOLVE($\lambda_p$)
    ENDWHILE
    $B \leftarrow B \cup \{(\lambda_p, L_p, V_p)\}$
  ENDWHILE
```

An interesting property of the search phase is that when an interval $[\lambda_\ell, \lambda_r]$ is resolved, i.e., yields a breakpoint, then the entire interval $[0, \lambda_r]$ has been resolved. Thus, if an a priori lower limit on L, say L_{min}, has been specified, the search phase can terminate as soon as an interval is resolved for which $L_r < L_{min}$. Upon termination of the algorithm, the set B can be used to construct the graphs of $V_T(0, \lambda)$ and $g(\lambda)$ as in Figures 2.1 and 2.2.

2.4 DISCUSSION

The key to efficiently solving the activity selection problem is efficiently solving the network dual subproblem, $D_T(0, \lambda)$, for each value of λ encountered in the search procedure. There are a number of very efficient network flow algorithms [6, 24, 26, 31], but only RNET [26] satisfies two requirements for use in this study: it provides facilities for parametrically varying the right hand side; and it is available at no charge for research purposes.

Not only does the solution to the activity selection problem prescribe a set of activities for on-block outfitting, it also provides information on the economic importance of each activity. From the primal problem, the parameters $(v_j - \lambda L_j)$ can be viewed as an indication of the relative absolute value of activity j, ignoring all other activities. Thus, if $(v_j - \lambda L_j)$ is negative, then activity j, considered by itself, is not a desirable activity for on-block outfitting.

Note, however, that even if $(v_j - \lambda L_j) < 0$, it may be desirable to choose activity j, because it is a predecessor of an activity k whose $v_k - \lambda L_k$ is very large. The relative or marginal value of activity j is indicated by the dual variable w_j.

If, in the activity scheduling problem, it appears that not all the selected activities can be scheduled within the given labor availability profile, then the values of $v_j - \lambda L_j$ and w_j can be used in deciding which activities to defer.

3.0 ACTIVITY SCHEDULING

The result of the activity selection algorithm is a set of on-block outfitting activities having the properties that:

(i) there is at least one outfitting window feasible schedule; and

(ii) the total labor required does not exceed the total available for the block, L.

To guarantee that this selection of activities is also resource profile feasible, it is necessary to exhibit a schedule that does not violate resource feasibility. Thus, the purpose of the activity scheduling procedure is to construct such a schedule if possible. In the event that not all the selected activities can be scheduled, the activity scheduling procedure should minimize the loss associated with the activities that must be deferred to on-board outfitting.

3.1 MATHEMATICAL MODEL

The notation developed in 2.1 will be used here, with the understanding that A, p(j), and s(j) refer only to activities that were selected for scheduling on-block. The algorithm developed for activity scheduling is a new variant of the classical SPAR-type heuristic [11, 39, 59], which is illustrated in Figure 3.1.

In the basic SPAR algorithm, "TIME" is the time of the next scheduling decision and is a set of <u>candidate</u> activities, i.e., activities whose predecessor activities have all been completed by the time the scheduling decision is required. R represents the <u>remaining</u> resource profile, i.e., the original resource profile adjusted for the activities already scheduled (some of which may still be in process at TIME). S is the set of activities chosen by the scheduling decision to be scheduled to begin at TIME, and

```
BEGIN SPAR
  TIME←0
  C←{1}
  WHILE $C \neq \phi$ DO
    SELECT ($R$, $C$, $S$)
    Update TIME
    Update $C$
    Update $R$
  ENDWHILE
END SPAR
```

Figure 3.1: Basic SPAR Heuristic Procedure

clearly, $S \subseteq C$.

In essence, the algorithm proceeds from the beginning to the end of the resource profile. At each point in time where a scheduling decision is required, the algorithm considers all those activities which are candidates for scheduling, and selects a schedule set from among them. The specific rule used for selecting the schedule set is what distinguishes the many SPAR-type heuristics.

Traditionally, SPAR-type algorithms have addressed only the following version of the resource constrained project scheduling problem. The resource is available at a given <u>constant</u> level and the objective is to minimize the project duration. The problem treated here is somewhat different, because a specific profile is given and the objective is to maximize the value of activities scheduled within the profile.

The SPAR-type algorithm is very myopic, in that when choosing S, it only considers information about the current activity attributes, such as slack (based on unconstrained CPM schedule calculations for activities not yet assigned a start time). Even the most sophisticated versions, e.g., [15, 37], do no more than simultaneously considering all of C by solving a knapsack problem [47, 48] to choose S.

In marked contrast, the algorithm presented here incorporates a limited look ahead feature in the process for choosing S. Thus, while no direct comparisons have yet been performed, it seems most likely that this algorithm can avoid, to some degree, the bad decisions that can arise from myopic scheduling rules.

The activity scheduling problem to be solved is considerably more difficult than the standard resource constrained project scheduling problem [11, 12, 15, 45] for two reasons:

(i) the resource profile, R, varies over the schedule horizon, and includes periods of declining availability;

(ii) the activity durations are also decision variables.

The second point is especially difficult to deal with. Note, for example, that since the durations are also decisions, the calculation of the CPM early and late start times is no longer simple and straightforward.

As mentioned above, the specific method for selecting the schedule set at each scheduling decision point is the primary element that distinguishes this algorithm from earlier SPAR-type heuristics. The details of the method developed in this research are given in the following section. The computer implementation is discussed in Chapter 4, and an evaluation of the performance on a set of test problems is presented in Chapter 6.

3.2 SCHEDULING DECISION RULE

The scheduling decision rule is the rule or procedure applied to determine $S \subseteq C$, the activities whose start times are assigned equal to TIME. The classical scheduling decision rule for SPAR-type heuristics is the following (see, e.g., [11, 12, 15]):

> Order the activities in C by some priority index, such as slack or late start time. Consider the activities in this order and select as many as can be accommodated by the currently available resources.

This scheduling rule can easily fail to obtain the fullest possible use of the resources available in the current time period. To overcome this deficiency, some authors have suggested using the priority index to construct an appropriate knapsack problem (see, e.g. [57]). This approach guarantees at least as good a solution, and possibly substantial improvement, with regard to current resource usage.

In these standard scheduling decision rules, two elements are important.

One is that *S* must be feasible with respect to current resources. The second is that the criterion in selecting *S* is to try to avoid delaying the project completion date. Thus, activities with zero slack are usually given absolute priority over other activities. The only inherent "value" associated with an activity is its criticality in delaying the project. In the classical problem, all activities will eventually be performed.

In scheduling the activities for on-block outfitting, the important concerns are somewhat different. Of course, the set *S* must still be feasible, not only considering current resource availability, but also future resource availability, since the resource profile declines at the end of the outfitting window. The criterion, however, is quite different. There is no project delay for the on-block outfitting activities - if an activity cannot be completed before the end of the outfitting window, it is simply deferred, and the associated opportunity cost is the savings foregone by not outfitting on-block.

The criterion in selecting *S* is therefore two-fold. First, complete utilization of the currently available resource is essential. Any unused resource in the current decision period implies that some activity cannot be completed within the given profile. Second, in selecting the activities to schedule now, it is essential that activities with zero slack, in the CPM sense, should be chosen; otherwise, one or more of the associated successor activities will have its early finish time pushed behond the end of the outfitting window.

Clearly, in the problem of on-block outfitting, the scheduling decision rule must accommodate a more complex set of issues than in the classical resource constrained project scheduling problem. Moreover, there is the added complexity of also having to determine the activity durations at the same time.

The scheduling decision rule developed for the on-block outfitting problem employs a three phase process. In phase one, a trial set, $\hat{S}$, is chosen. The trial set is guaranteed to be feasible with respect to current resource availability. The criterion used in this trial selection is to minimize the maximum delay for deferred activities, considering the next possible schedule decision time, the resources that will be available then and the resources required by the deferred activities.

The second phase of the process is not invoked unless the trial selection causes a resource violation at some future time period. In this case, the activity durations are modified, if possible, and, if necessary, a gradual penalty method is iteratively applied to the selection algorithm. The method guarantees that a resource feasible selection will be obtained.

At the end of phase two (or phase one if the second phase is not required) a final selection, S, has been made. Phase three of the scheduling decision rule assigns durations to the activities in S. The goal of this phase is to completely utilize the currently available resource, without introducing any infeasibilities in future periods. An outline of procedure SELECT (R, C, S) is given in Figure 3.2.

3.2.1 Trial Selection of S

A key element in the trial selection is insuring resource feasibility in the current time period. Note, however, that unless durations have been specified, the resource rate for each activity in C is not known, except as a range. Thus, the first step in trial selection is to assign tentative activity durations to those activities which have not yet been scheduled.

```
PROCEDURE SELECT ($R$, $C$, $S$)
  IF (no activity can be selected) THEN
    $S \leftarrow \emptyset$
    RETURN
  ENDIF
  IF (every activity can be selected) THEN
    Set Durations
    RETURN
  ENDIF
  Determine Criterion Coefficients              }
                                                } Phase I
  Solve Knapsack Problem to Obtain $\hat{S}$    }
  IF (no resource violation) THEN               }
    Set Durations                               } Phase III
    RETURN                                      }
  ENDIF
  WHILE (resource violation by $\hat{S}$)       }
    Compute Penalty                             } Phase II
    Solve Knapsack Problem for New $\hat{S}$    }
  ENDWHILE                                      }
  Set Durations                                 } Phase III
  RETURN                                        }
END PROCEDURE
```

Figure 3.2: Outline of Scheduling Decision Rule

The assignment of tentative durations could be made by any one of a vast number of rules. The procedure adopted here is the simple expedient of assigning:

(1) maximum durations to candidate activities, and

(2) minimum durations to remaining unscheduled activities.

This rule is easy to implement and allows the largest possible number of candidates to be selected for scheduling. Note that it may be necessary to modify the maximum duration, if the maximum duration would prevent the activity from being completed by time TMAX.

With these tentative durations, the resource usage rates for candidate activities are computed by dividing total labor required by the tentative durations. Denote the resulting rates by r_j. The selection, S, must satisfy

$$\sum_{j \varepsilon S} r_j \leq R_0$$

where R_0 is the amount of uncommitted resource in the current period. Also, if the resource profile declines in the t^{th} period hence, the selection must also satisfy

$$\sum_{\substack{j \varepsilon S \\ d_j > t}} r_j \leq R_t$$

where R_t is amount of uncommitted resource in the t^{th} period from now, and d_j is the tentative duration for activity j.

There are clearly two cases in which the selection is easily resolved. If $r_j > R_0$ for all $j \varepsilon C$, then none of the current candidates can be selected, so $S = \emptyset$. Also, if

$$\sum_{j \varepsilon C} r_j \leq R_0$$

and

$$\sum_{j \varepsilon C} r_j \leq R_t$$

$$d_j > t$$

then all the candidate activities can be selected, so $S = EC$.

If neither case occurs, then a specific selection rule is required. Observe that if a value, $\bar{v}_j$, is associated with $j \varepsilon C$, then the set S can be determined by solving a knapsack problem [ ]:

$$(KP_0) \qquad \text{maximize} \quad \sum_{j \varepsilon C} \bar{v}_j x_j$$

$$\text{s.t.} \quad \sum_{j \varepsilon C} r_j x_j \leq R_0$$

$$x_j \ \varepsilon \ \{0, 1\}$$

Of course, the solution must also be tested against the future period resource constraint.

In the procedure implemented here, the value is determined as a weighted sum of two quantities. One of the quantities is simply the savings associated with on-block outfitting as opposed to on-board outfitting. The second quantity is referred to as the "minimum delay" associated with failing to schedule the activity now.

If the activity is not scheduled now, it cannot possibly be scheduled until the next schedule decision period. The next time when a scheduling decision will be required is when either a previously scheduled activity finishes, thus releasing resources, or when the resource profile increases. Thus, it is possible to look ahead and determine a lower bound on the time to the next scheduling decision, by assuming that the shortest activity in C will be selected. (Similarly, an upper bound can be determined by ignoring the activities in E. Let τ be the soonest that the next scheduling decision will be possible.

Suppose that the CPM late start times LS(j) have been calculated, based on the tentative activity durations. If for some $j \varepsilon C$, $\tau > LS(j)$, then failing to schedule j now means that some outfitting activities must be deferred. Thus, the quantity $\tau - LS(j)$ is a measure of the importance of scheduling activity j in the current period.

The values, $\bar{v}_j$, are computed by:

$$\bar{v}_j = \alpha \cdot \max (0, \tau - LS(j)) + \beta v_j$$

where values for α and β are to be specified and v_j is as defined earlier. The appropriate values for α and β could depend on the type of problem, and should be based on experimentation.

3.2.2 Future Resource Feasibility

Because the knapsack problem (KP_0) ignores the resource constraints in future periods, it may violate them. In this event, a two-step procedure is applied to force resource feasibility. First, the activity durations are modified. If an activity, $j \varepsilon C$, has a minimum duration which does not exceed τ-TIME, then the tentative duration is made equal to the lessor of those two. This compresses the activity so that, if scheduled now, it will be completed before the period which contained the future resource violation.

If the activity's minimum duration exceeds τ-TIME, then the tentative duration is made equal to the maximum duration. The rationale is to stretch out the activity, thus reducing the associated rate of resource usage. With these modified durations, the knapsack problem (KP_0) is reformulated and solved. If the resulting selection is feasible in future time periods, then the scheduling decision rule proceeds to phase three.

If the resulting selection is not feasible, then (KP_0) is used as the basis for a gradual penalty method which eventually must guarantee a feasible selection. The penalty is applied as follows. Let C_τ be the set of candidate activities whose tentative durations exceed τ-TIME, i.e., if selected, they will be in process during the period when the resource violation occurs. The values of these activities are modified by

$$\bar{v}_j \leftarrow \bar{v}_j - \lambda \qquad j \in C_\tau$$

where initially λ is small.

Suppose λ is initialized to λ_1, and let (KP_1) be the associated knapsack problem. If (KP_1) is not feasible at time τ, then increase λ to $\lambda_2 > \lambda_1$. As this process is repeated, sooner or later, λ will be large enough so that $\bar{v}_j < 0$ for $j \in C_\tau$, so that activity j is not selected. Thus, feasibility is guaranteed. On iteration k of this gradual penalty method, λ can be computed by

$$\lambda^k = \delta 2^k$$

where δ is a parameter to be determined experimentally.

3.2.3 Assigning Durations

Once the set S has been determined, it may be possible to reduce some activity durations without violating feasibility. If so, this should be done to improve resource utilization. The method used here is to sort S by decreasing values of $\tau - LS(j)$. Then consider $j \varepsilon S$ and determine if d_j can be reduced by one unit. If so, then d_j is reduced, and S is resorted. If d_j cannot be reduced, then the next activity is considered, etc. The process halts when all activities in the sorted set have been considered without reducing any durations.

3.3 DISCUSSION

As with many heuristics, this one has a strong flavor of ad hoc decision making in its design. Many of the design decisions can be justified logically, but in many cases, only empirical evidence can justify the deeision. A limited amount of such evidence is offered in Chapter six.

It should be noted that the decision rule could be made substantially less ad hoc by making the activity duration decision part of the selection decision. Suppose $j \varepsilon C$ has several possible durations, d_{jt}, and for each duration, a resource usage rate, r_{jt}. Also, suppose that a duration-specific value can be assigned, $\bar{v}_{jt}$. Then the selection decision can be made by a multiple choice knapsack problem [48] as follows:

$$(\text{MCKP}_0) \qquad \max \sum_{j \varepsilon C} \sum_t \bar{v}_{jt} x_{jt}$$

$$\sum_{j \varepsilon C} \sum_t r_{jt} x_{jt} \leq R_0$$

$$\sum_t x_{jt} \leq 1 \qquad j \varepsilon C$$

$$x_{jt} \varepsilon \{0, 1\}$$

This would be, conceptually, a superior way to determine both S and the activity durations. A variant of the previous gradual penalty method could still be used to guarantee future resource feasibility. The obvious question is, "Why wasn't this approach used?" The answer is straightforward. ($MCKP_0$) is substantially more difficult to solve, and there is no readily available software for its solution. In contrast, (KP_0) is quite easy to solve, and a good procedure is widely available [47].

Because of the limited scope of this research project, it was decided to focus on developing a methodology that could be implemented and tested using the available software. At the same time, the methodology is flexible enough to accommodate the more powerful technique, should software for it become available.

4.0 COMPUTER IMPLEMENTATION

The purpose of this chapter is to give an overview of the software implementation of the activity selection and scheduling methodologies developed in Chapters 2 and 3. The structure of the computer codes will be discussed along with the computing environment necessary to support their use.

The methodology has been implemented as three distinct computer programs. The first, SELECT, operates on the problem data plus a user provided outfitting window duration. SELECT produces a table listing resource allocations and associated cost savings and Lagrangian multipliers.

A resource allocation, cost saving, and corresponding Lagrangian multiplier, are selected from this table and used as input, along with the original problem data and the outfitting window, to the second program EXTRACT. This program generates a modified problem, by extracting those activities chosen in program SELECT for the given resource allocation and outfitting window. In addition, program EXTRACT computes several quantities for each activity in the modified problem. These quantities are included in the modified problem data set.

The modified problem data set is the input to program SCHEDULE, the third, and last, of the computer programs. Program SCHEDULE determines a starting time for each activity so that both precedence requirements and resource availability are satisfied. The output from SCHEDULE consists of a summary indicating which activities, if any, could not be scheduled within the available resources. An optional output is the detailed activity schedule and the associated period-by-period labor requirement. Figure 4.1 summarizes these operations.

All programs were written in FORTRAN, ANSI77, and extensive use was

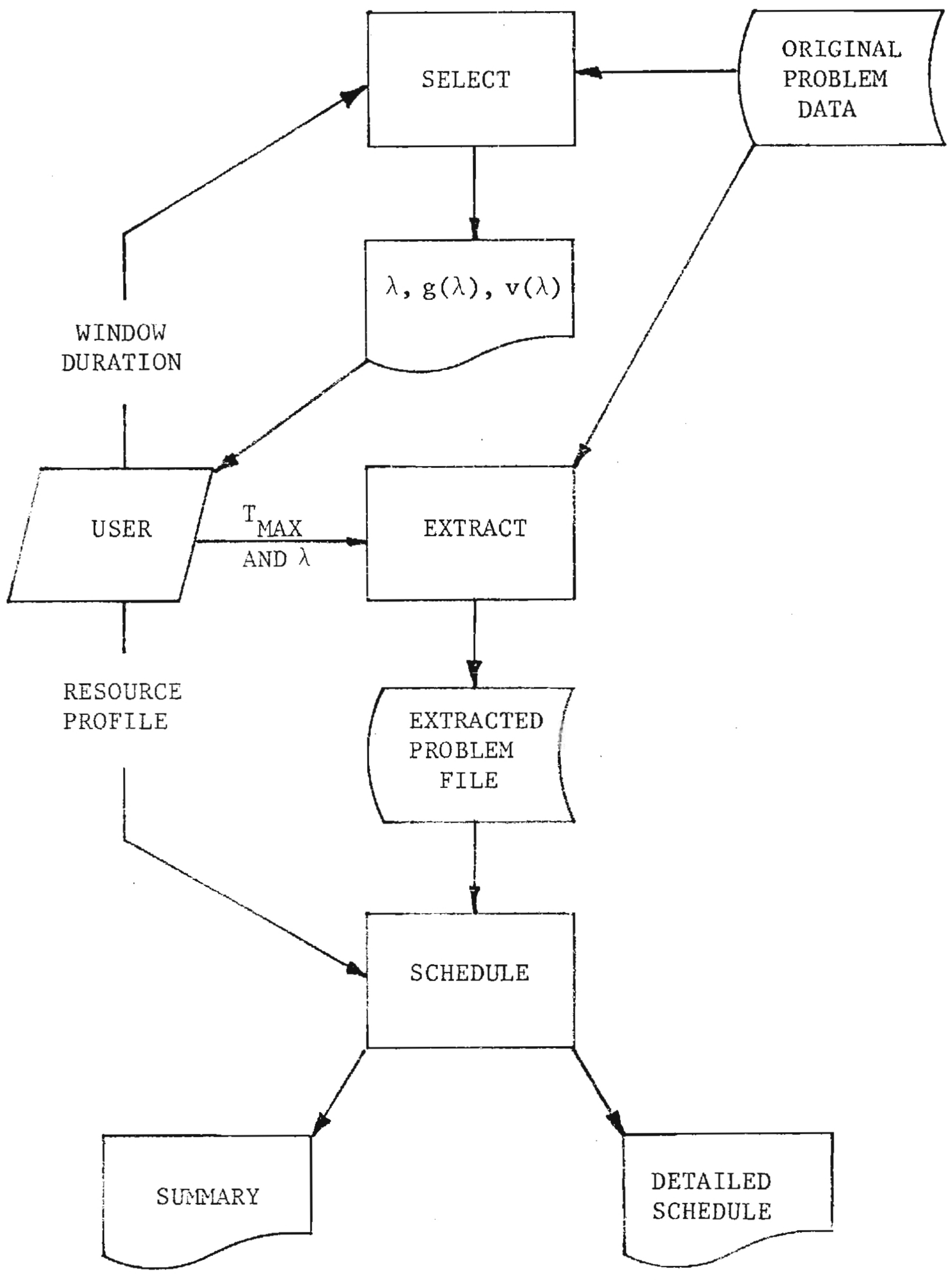

Figure 4.1: Implementation of Solution Methodology

made of the block-IF structure. The codes are substantially internally documented. The compiler used was the FTN5 compiler under the NOS operating system of the CDC CYBER 74. The programs are designed for execution in interactive batch mode, with output restricted to eighty columns to facilitate viewing from CRT's.

4.1 PROGRAM SELECT

Figure 4.2 illustrates the program flow and Table 4.1 lists the program elements and their functions. All of the program elements, except RNET [26] were developed as part of this research effort. RNET is a proprietary code for solving various network flow optimization problems. RNET Version 3.61 was used without modification, and is available from Rutgers University. Because of the proprietary nature of RNET, it is not included in the code listings provided here.

As configured presently, SELECT can accommodate problems with up to 250 activities and 1000 precedence relations. The array requirement is 19,250 words, or 17N + 6P + 9B words where

N = number of activities, including dummies

P = number of precedence relations

B = overestimate of the number of breakpoints

B was initially set at 1000, but it can be shown that the number of breakpoints will never exceed N. Thus, the program could have been configured to require only 13,000 words for array storage.

The source code for program SELECT, exclusive of the RNET package, contains approximately 700 lines, and compiles to approximately 1200 words of object code. Thus, program SELECT does not require a large scale main frame computer environment, although that was the environment in which it was developed. A complete source listing of SELECT is contained in

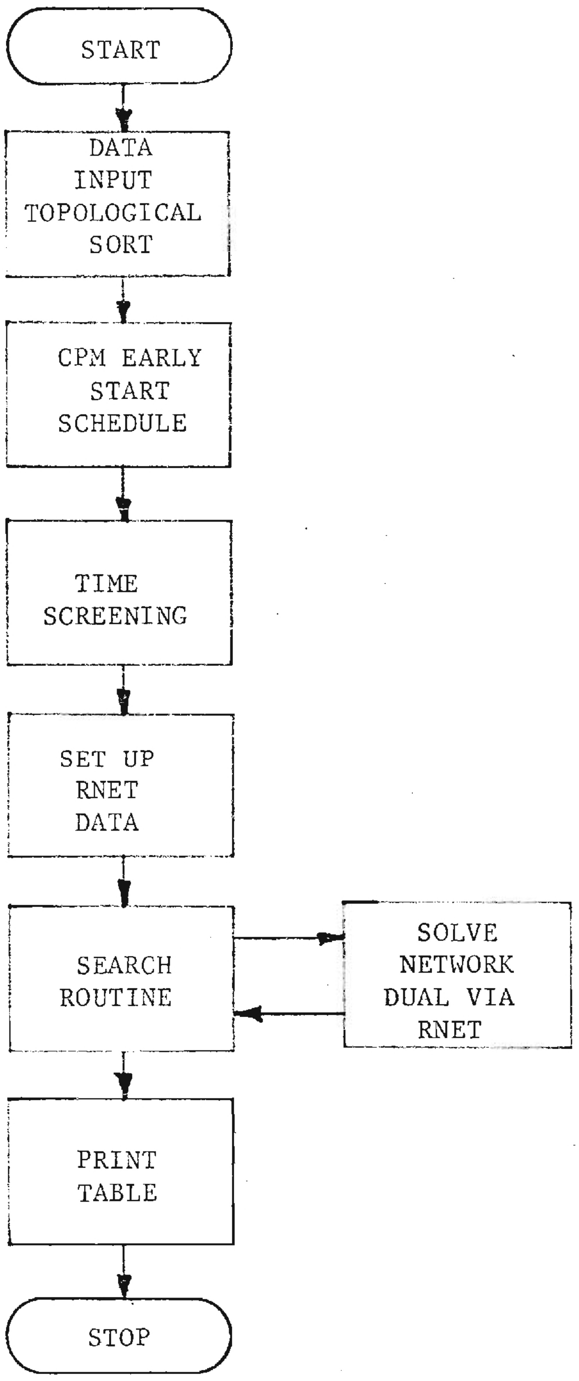

Figure 4.2: SELECT Program Flow

Table 4.1: SELECT Program Elements

SELECT:	main program; controls program flow, performs data input, conversion, and output
DATIN:	subroutine subprogram; reads problem data from disc file; called from SELECT
TSORT:	subroutine subprogram; creates a topological ordering of the activities; called from SELECT
EARLY:	subroutine subprogram; determines CPM early start schedule; called from SELECT
SOLVE:	subroutine subprogram; solves the Lagrangian relaxation for given multiplier; called from SELECT
PROJECT:	function subprogram; used in search routine to determine next multiplier; called from SELECT
PUSH:	subroutine subprogram; add a record to a stack; called from SELECT
POP:	subroutine subprogram; get a record from a stack; called from SELECT
ERRSTP:	subroutine subprogram; provides error exit for all detected abnormal terminations; called from SELECT, PUSH, POP, and SOLVE
RNET:	suite of subroutine subprograms; provides solution to network dual problem; called from SOLVE

4.2 PROGRAM EXTRACT

Program EXTRACT is basically the same as program SELECT, except that no search routine is required, since the multiplier is specified. EXTRACT does perform one function that is not part of SELECT - it generates a modified problem file, containing data for only those activities contained in the optimal solution to the activity selection problem.

Because of its similarity to SELECT, no listing for EXTRACT is included in this report. Listings are available, on request, from the School of Industrial and Systems Engineering, at Georgia Tech.

4.3 PROGRAM SCHEDULE

Figure 4.3 illustrates the program flow and Table 4.2 lists the program elements and their functions. All of the program elements except KNAP and SORT6 [42] were developed as part of this research effort. KNAP and SORT6 are proprietary codes and are available from R. Nans for a small fee. Since both were used without modification, they are not included in the listings provided here.

As configured, SCHEDULE can accommodate problems with up to 250 activities and 1000 precedence relations. It is assumed that the resource profile has at most 25 distinct intervals, and that it declines at most twice. The array requirement is 7875 words, or 23N + 2A + 75 words, where

N = number of activities, including dummies

A = number of precedences.

Figure 4.4 illustrates the hierarchical structure of SCHEDULE. It should be noted that the design of the software for SCHEDULE is quite

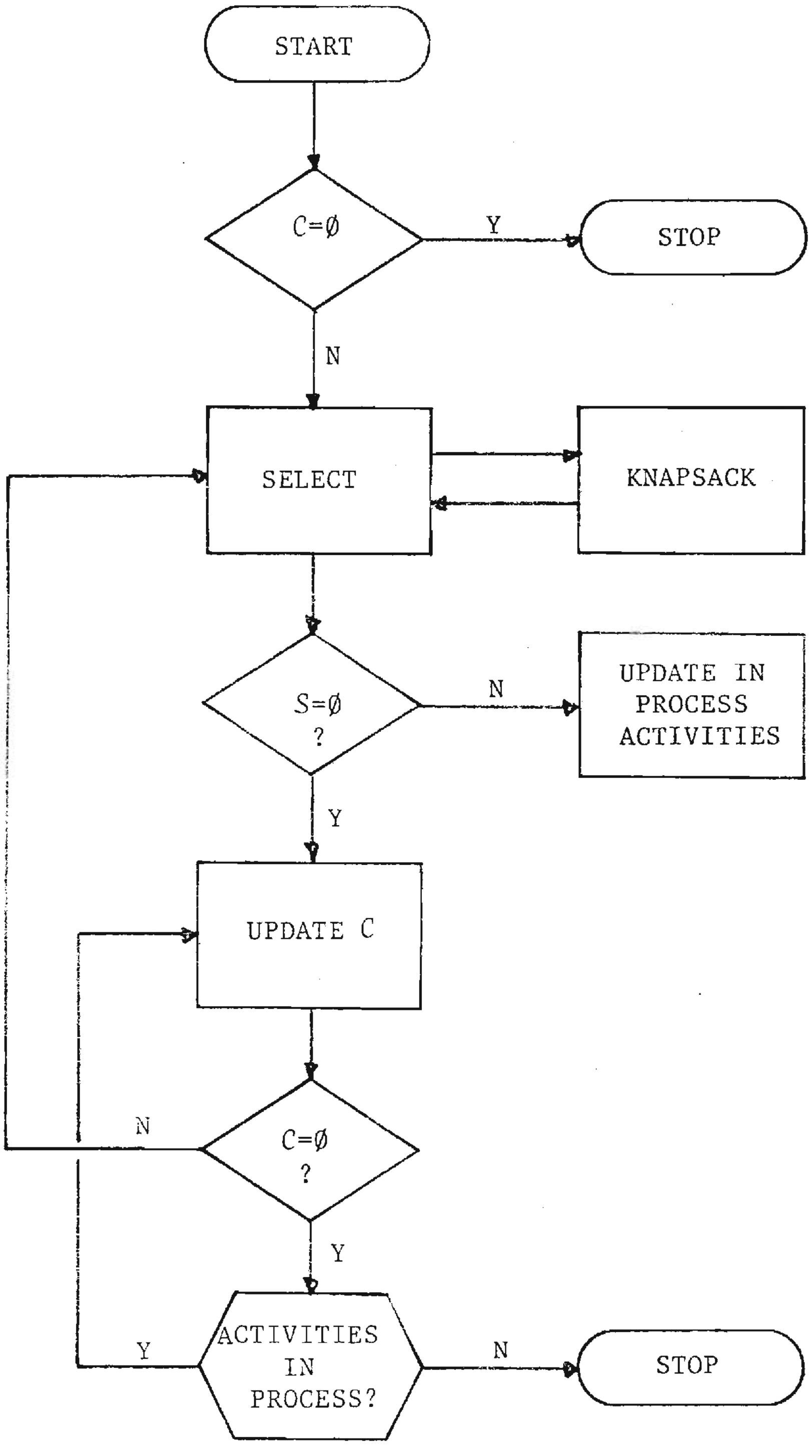

Figure 4.3: SCHEDULE Program Flow

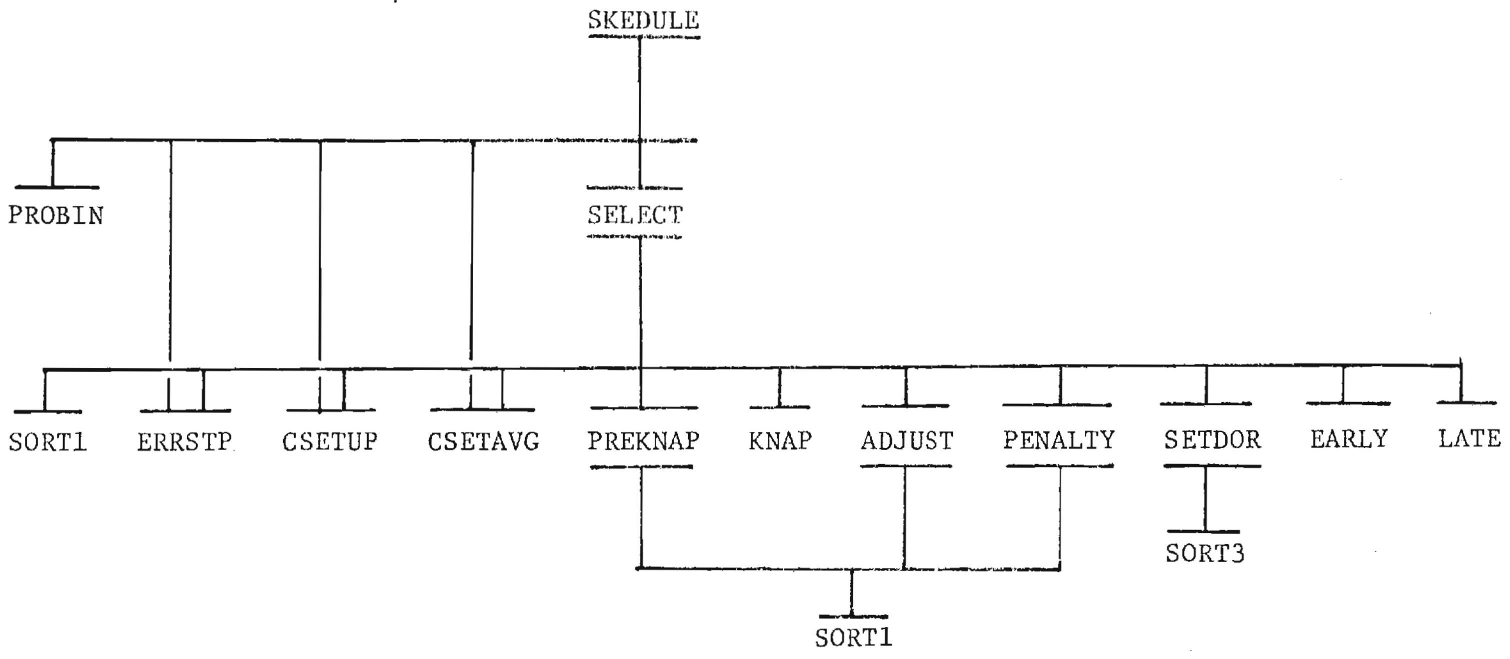

Figure 4.4: Hierarchical Structure of SCHEDULE

Table 4.2: SCHEDULE Program Elements

SKEDULE:	main program; controls program flow, performs data input, output, and interface monitor
PROBIN:	subroutine subprogram; reads modified problem data from disc file; called from SKEDULE
CSETUP:	subroutine subprogram; updates active node list for activities selected by scheduling rule; called from SKEDULE and SELECT
CSETAVG:	subroutine subprogram; updates candidate list by examining active nodes that have just completed; called from SKEDULE and SELECT
SELECT:	subroutine subprogram; performs scheduling decision rule; called from SKEDULE
ERRSTP:	subroutine subprogram; manages all detected run-time errors; called from various routines
EARLY:	subroutine subprogram; calculates CPM early start schedule; called from SELECT
LATE:	subroutine subprogram; calculates CPM late start schedule; called from SELECT
PREKNAP:	subroutine subprogram; computes objective function coefficients for selection problem; called from SELECT
KNAP:	subroutine subprogram; solves binary knapsack problem; called from SELECT
ADJUST:	subroutine subprogram; modifies activity durations in (KP_0) is not feasible; called from SELECT
PENALTY:	subroutine subprogram; modifies knapsack coefficients in the gradual penalty method; called from SELECT
SETDUR:	subroutine subprogram; determines durations for selected activities; called from SELECT
SORT1:	subroutine subprogram; sorts a set into increasing order; called from SELECT
SORT6:	subroutine subprogram; determines "bang-for-buck" ordering of knapsack variables; called from PREKNAP, ADJUST, PENALTY
SORT3:	subroutine subprogram; sorts a set into decreasing order of an auxiliary value; called from SETDUR

flexible. Not only does it permit various rules to be tried for the on-block scheduling problem, but the same basic structure can be used as the basis for algorithms for other, more traditional project scheduling problems.

The source code for SCHEDULE, exclusive of KNAP and SORT6, contains approximately 1025 lines, and the complete program compiles to approximately 4400 words of object code. As with SELECT and EXTRACT, SCHEDULE could quite easily be implemented on a minicomputer. The source listing is contained in Appendix B.

4.4 DISCUSSION

A substantial development effort is contained in the computer codes discussed in this chapter. These codes are designed to be easy to understand and modify, flexible, and efficient. In order to achieve these goals, state-of-the-art techniques in data structures (such as linked-lists, etc.) are employed along with top-down, structured program design. The codes were designed and developed with the intention that they would provide a basis for continuing investigation of this and other related project scheduling problems.

It remains true, however, that these are <u>experimental</u> codes. They do not embody sophisticated user interfaces or error trapping routines, nor do they utilize sophisticated data base techniques. Thus, they are <u>not</u> likely candidates for immediate application in shipyards. They very well might, however, be used to prototype an actual shipyard system.

5.0 GENERATION OF TEST PROBLEMS

No shipyard, to the best of our knowledge, currently plans outfitting in the manner suggested by our model. Therefore, the information regarding network structures and activity parameters were not available from shipyards in a directly useable form. Fortunately, one major shipbuilder, who required anonymity, was generous enough to provide us with actual planning and performance information on the construction of a large product carrier.

Although this data was not directly useable, it permitted us to estimate reasonable values for parameters such as crew size and labor content per work package. It also provided an example of a work package precedence network.

Based on this information, a process was developed for generating realistic test problems. Both the network structure and specific activity parameters in these generated problems reflect a degree of randomness introduced by Monte Carlo sampling techniques. The sampling procedures themselves incorporate the information developed from the actual product carrier case.

The test problem generation process was coded in FORTRAN for use under the FTNTS subsystem of the NOS operating system for the CDC CYBER computer. The program allows a wide range of flexibility in the number of elements, units and precedence relationships, and also allows flexibility in the specification of average work package size and duration and average crew size. The generation procedure is described in greater detail below.

5.1 NETWORK CONSTRUCTION

The outfit activities for a hypothetical block are depicted in the network of Figure 5.1. A set of elements involves nodes 1 through 7 and

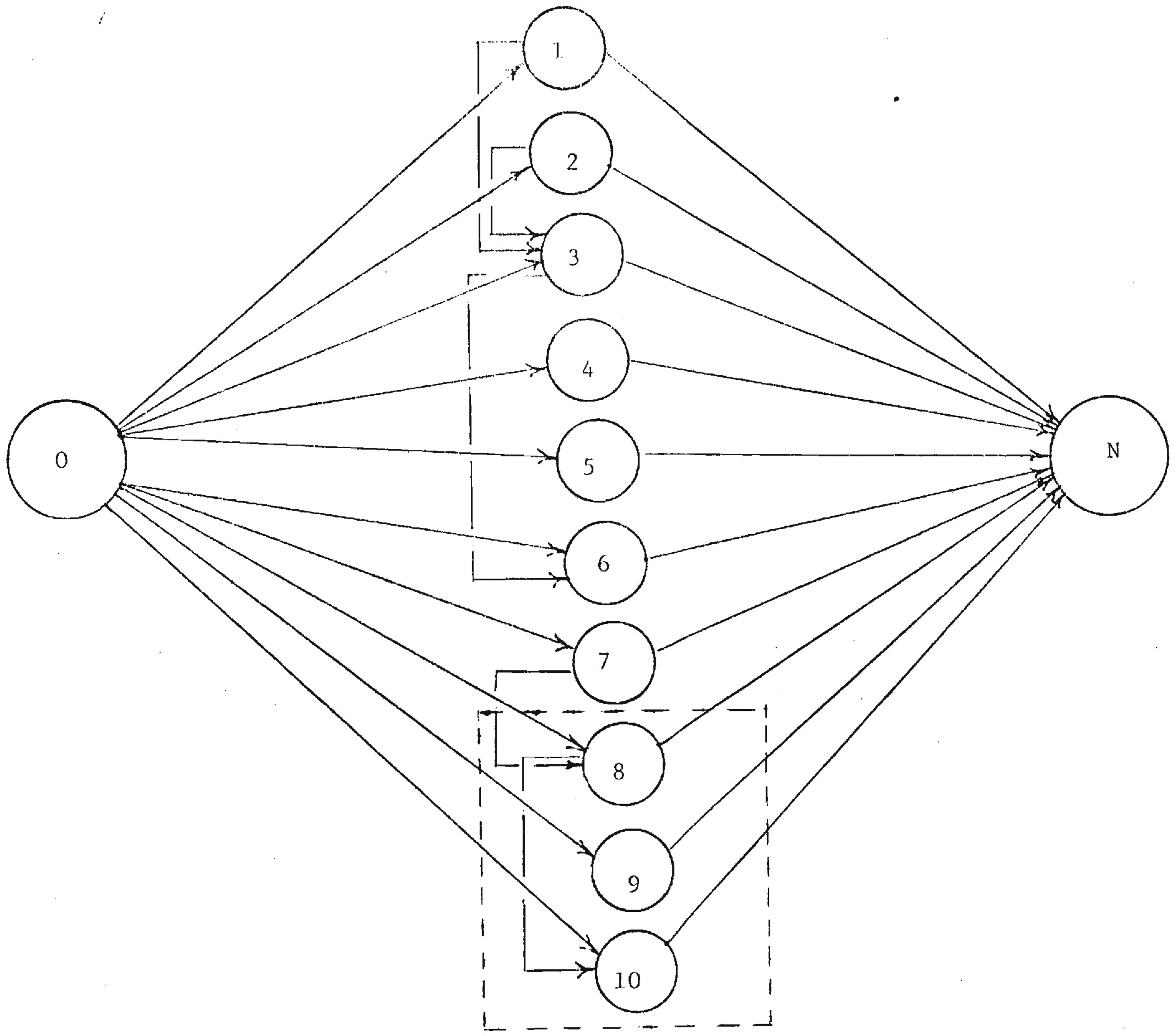

Figure 5.1: Hypothetical On-Block Outfit Activity Network

the set of units is depicted by nodes 8 through 10. In the absence of constraints and assuming that it is always advantageous to outfit on-block compared to on-board, all of these outfitting activities, i.e. nodes 1 through 10, would be selected for on-block outfitting. When constraints come to bear or the savings are reversed, those appropriate outfitting activities may be deferred to the on-board mode.

Another notable feature of Figure 5.1 is the arc representation of technological order for precedence among selected on-block work elements. An arc from node 3 to node 6 is representative of this feature and implies that the outfit element corresponding to node 3 must be completed prior to starting the element associated with node 6. The density of these arcs is controlled by the user of the test problem generator, although default parameters are set. The user specifies a value, PERC1, which is a percentage of the number of nodes that correspond to outfitting elements. The resulting number is the number of precedence arcs that will be generated between outfitting elements. In Figure 5.1, PERC1 = 0.428 gives 3 precedence arcs. PERC1 is also applied to the set of outfitting units shown within the dotted lines on Figure 5.1. PERC1 = 0.428 gives one arc within this set. A second factor, PERC2, is a percentage of the total number of nodes, i.e. both outfitting elements and outfitting units. The number resulting from the use of PERC2 gives the arcs between members of the outfit element set and members of the outfit unit set. PERC2 = 0.10 gives one such arc in Figure 5.1.

Each of these outfitting activities has information associated with it as described in Chapter 2. This information must be generated for the test problems and then stored in a data file.

5.2 DATA REQUIREMENTS

The network may be depicted graphically as in Figure 5.1 or depicted in a data base for computer implementation. The information required to describe the activities and to constrain the choices with respect to outfit mode for the activities is also to be included in the data base. Thus, there are essentially two types of data: activity data and precedence data. These data may be characterized in the form of two data files whose record contents are as indicated below:

Activity File:

- Activity Identifier
- Labor Man-hours
- Duration Interval (d_{min}, d_{max})
- Pointer to Precedence File

Precedence File:

- Number of Successor Activities
- Number of Predecessor Activities
- List of Successor Activity Numbers
- List of Predecessor Activity Numbers

Note that on each file, there will be one record for each node in the network, i.e., one record per outfitting activity.

The contents of these files can be demonstrated using node number 6 as an example.

Activity File	Precedence File (176)
6	1
631	2
7, 11	N
176	0
	3

In this case, activity number 6 is estimated at 631 man-hours with a minimum duration of 7 time periods and a maximum duration of 11 time periods. Precedence File record 176 contains the precedence information.

In the Precedence File at record 176, there is one successor to activity 6, namely activity N. There are 2 predecessors to activity 6, namely activities 0 and 3.

5.3 PARAMETER ESTIMATION AND DATA GENERATION

The process of generating data for six test problems or hypothetical blocks is one that necessarily involves the transformation of certain data from the product carrier case in a way that respects the network structure described earlier. The initial information from the product carrier data are the planned <u>duration</u> and the <u>total</u> <u>man-hours</u> for each member of the outfit activity set. All other data for each hypothetical case are either derived from this initial information or generated arbitrarily in a manner consistent with the circumstances involved.

For the elements of the outfit activity set, it is only necessary to calculate a minimum duration, d_{min}, and a maximum duration, d_{max}. It was assumed that the difference between d_{max} and d_{min} would not exceed two time periods for the cases where the planned duration is less than or equal to five time periods and not exceed four time periods for a planned duration of over five periods. Considering this planned duration as an average duration, d_{avg}, the following provides the values of d_{min} and d_{max}:

$$d_{min} = \begin{cases} d_{avg} - 1\ , & \text{where } d_{avg} \leq 5 \\ d_{avg} - 2\ , & \text{where } d_{avg} > 5 \end{cases}$$

$$d_{max} = \begin{cases} d_{avg} + 1\ , & \text{where } d_{avg} \leq 5 \\ d_{avg} + 2\ , & \text{where } d_{avg} > 5 \end{cases}$$

A fundamental assumption about outfit planning is that outfitting should always be done as early as possible in the production process. Thus, early outfitting is generally attractive from the standpoint of cost reduction and the imposition of various resource constraints, such as available man-hours, requires outfitting to move away from this desired plan. The generated test problems respect this assumption by having the labor man-hours for an outfitting activity in the on-block mode to be less than the corresponding man-hours in the on-board mode, hence a 'savings' is generated for each on-block outfitting activity.

This savings is expressed in labor dollars for each outfitting activity in the test problems. Further, it is randomized as follows:

(1) $$\text{Labor Savings (LS)} = (\text{Man-Hours}_{\text{Board}} - \text{Man-Hours}_{\text{Block}}) \times (\$15 \text{ per Man-Hour}) \times (\text{Random Number})$$

where $\text{Man-Hours}_{\text{Board}}$ and $\text{Man-Hours}_{\text{Block}}$ are the labor man-hours required for that outfitting activity if it is performed in an on-board or an on-block mode, and Random Number is sampled from a $U(0.5,\ 1.5)$ distribution.

The randomization causes some savings to be greater than others which reflects the likely situation facing outfit planners.

The use of <u>actual</u> shipyard data within a <u>hypothetical</u> situation involving constraining conditions, such as labor man-hours associated with outfit modes, requires several transformation processes. One necessary transformation process is that of man-hours, d_{min} and d_{max}, for the network elements if they were done in an on-board mode as well as the corresponding on-block mode data. The important transformation term in this process is the crew size (CS). Assuming each crew member works an eight hour day, then crew size for these on-board and on-block activities is determined by (2).

(2) $$\text{Crew Size (CS)} = \text{Total Man-Hours} \div (d_{avg} \times 8 \text{ hours})$$

Using the product carrier data for activity man-hours, the CS values were observed to vary widely, but the range from 8.33 to 9.00 covered a high percentage of the activity crew sizes. This was then used as the distribution range from which crew sizes were randomly drawn for the test problem outfitting activities. In keeping with the assumption stated earlier, the crew size was viewed as increasing when considering the on-board outfit mode and hence a similar process was followed for on-board outfitting activities with a CS range of 9.38 to 10.18.

The crew size range, duration and man-hours interact together. With crew size range already generated from the baseline ship data, the man-hour estimates were fixed and these values used jointly to establish durations. The man-hour estimates were constrained to not exceed 2500 man-hours for any outfitting activity and to increase from those values associated with outfitting on-block up to the outfitting on-board mode.

Finally, the values for d_{min} and d_{max} for each network node were generated using (3) and these appear in column 2 of Table 5.1.

(3) $$d_{min} = \text{Man-Hours} \div 8 \div \text{Maximum Crew Size}$$
$$d_{max} = \text{Man-Hours} \div 8 \div \text{Minimum Crew Size}$$

5.4 SUMMARY

The test problem generator is a powerful tool for evaluating outfit planning methods. It requires only typical values for certain activity parameters, thus avoids many of the difficulties associated with proprietary

Table 5.1: Test Problem Parameters

	(2) d_{min}, d_{max}	(3) Range of Crew Size	(4) Range of Man-Hours
Outfitting Elements on-Block	7, 9	8.33-9	467-648
Outfitting Units on-Block	10, 12	8.33-9	666-864
Outfitting Elements on-Board	12, 14	9.38-10.18	900-1140
Outfitting Units on-Board	16, 18	9.38-10.18	1200-1466

data. The generator program is small (less than 300 lines of code), and written in FORTRAN so it is easily transportable. Thus, the same problems can be solved by different, competing methods. Finally, since the generation process is fairly fast, a wide range of problems can be presented in order to test the limits of a proposed methodology.

6.0 COMPUTATIONAL EVALUATION

This chapter presents a small example problem to illustrate the methodology and the computer programs, and then presents computational results for some larger problems.

In developing an evaluation of a decision aiding technique, two types of measures are usually important. First, how well does the technique perform with regard to the solutions it proposes? Are they optimal or good solutions? In the present analysis, this question is extremely difficult to answer, for there is no standard of comparison.

The best that can be done in this situation is to provide an absolute evaluation of the solutions. For the results to be presented, this evaluation takes the form of several performance ratios, specifically, available resource utilization, and selected savings realization. If these two indices are always large (close to one) then the solutions are ruled "good." On the other hand, if the ratios are always low (near zero) then the solutions are not very good.

The second important measure is the expense of using the tool. That is, how much manual effort is required, and what computational resources are consumed (primarily main memory and computation time). As indicated in Chapter four, the procedures being evaluated require only modest amounts of memory, and could, in fact, be run on minicomputers that support FORTRAN.

The question of analyst time and computation time are a bit harder to answer. We will not even address the former, since it depends on so many parameters. For computation time, the exact figures, measured by a real-time cpu clock, are reported for the scheduling program. For the selection program, rough estimates are given for the average time to generate one table, such as the one in Table 6.2.

6.1 EXAMPLE PROBLEM

A problem with 10 outfitting elements and 3 units was created using the problem generator. The data for this small example is given in Table 6.1 and Figure 6.1 displays the precedence network. The sample problem was analyzed using SELECT, to determine an appropriate window duration and resource level. Table 6.2 presents a sample analysis for an outfitting window of 18 time periods.

A λ value of 0.61 was chosen from Table 6.2, corresponding to 591 man-days of outfitting and a savings of $65880. The outfitting window of 18 and λ of 61, along with the original problem data, were the input to EXTRACT. The data for the resulting modified problem are given in Table 6.3 and the corresponding precedence network is shown in Figure 6.2.

The sample problem given in Table 6.3 and Figure 6.2 was next solved using SCHEDULE. The resource profile used is shown in Figure 6.3. Note that the total length of the profile is 25 time periods, which is less than the 18 time periods used earlier. This reflects the fact that it is usually the labor availability, rather than strictly technological precedence, which determines the time required to complete the activities.

Table 6.4 presents the summary report from SCHEDULE. Not all the selected activities could be scheduled within the given profile, which is not unexpected. This is a small problem, and the level of resources available is small, resulting in some resources being "unusable." In practice, the response to this solution could be to modify the profile, modify the set of activities (using SELECT again) or to accept the current solution and simply return the unused resources to a central resource pool.

Table 6.1: Sample Random Problem Data[+]

Activity Number	Total Labor	Savings	Duration Min	Duration Max
1	606	10222	7	11
2	574	6244	6	10
3	494	6240	5	9
4	538	7312	6	10
5	625	6899	7	11
6	621	4637	7	11
7	539	10222	6	10
8	624	3953	7	11
9	544	11346	6	10
10	574	8061	6	10
11	740	4719	9	13
12	824	12106	10	14
13	811	10685	10	14

[+]By convention, the outfit units have the largest activity numbers.

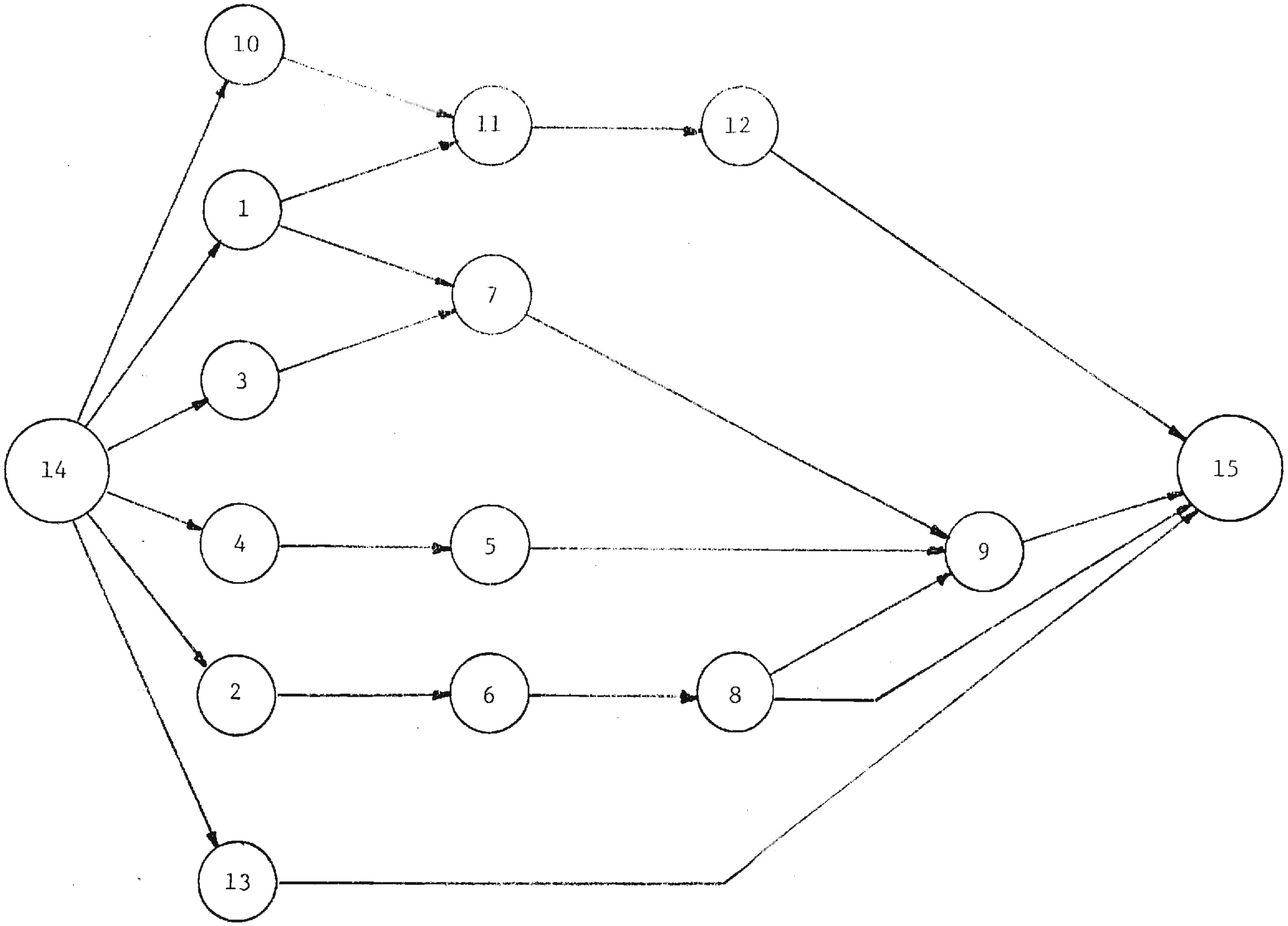

Figure 6.1: Sample Random Problem

Table 6.2: Sample Problem Analysis

```
                               PROBLEM SUMMARY STATISTICS
                            NUMBER OF ELEMENTS:           10
                            NUMBER OF UNITS:               3
                            MINIMUM CPM DURATION:         26
                            MAXIMUM CPM DURATION:         42
                            TOTAL LABOR:                1009
                            TOTAL SAVINGS:            102637

 WHAT VALUE FOR TEND
? 16
1      BREAKPOINTS OF V( 16,LAMBDA) AND G(LAMBDA)

       LAMBDA            V( 16,LAMBDA)          G(LAMBDA)           VALUE
       ------            -------------          ---------           -----


    0.                     .752340E+05          .760000E+03       .752340E+05
     .512827E+02           .362592E+05          .668000E+03       .705160E+05
     .602079E+02           .302971E+05          .591000E+03       .658800E+05
     .879297E+02           .139136E+05          .520000E+03       .596370E+05
     .884468E+02           .136430E+05          .442000E+03       .527360E+05
     .105782E+03           .598224E+04          .341000E+03       .420540E+05
     .109120E+03           .484420E+04          .274000E+03       .347430E+05
     .113521E+03           .383819E+04          .203000E+03       .268830E+05
     .128692E+03           .570075E+03          .750000E+02       .102220E+05
     .130293E+03          0.                   0.                0.
```

Table 6.3: Sample Random Problem Data - Modified

Activity Number	Original Activity	Total Labor[+]	Savings	Duration	
				Min	Max
1	14	0	0	0	0
2	1	75	10222	7	11
3	2	71	6244	6	10
4	3	61	6240	5	9
5	4	67	7312	6	10
6	10	71	8061	6	10
7*	13	101	10685	10	14
8	7	67	10222	6	10
9	5	77	6899	7	11
10	---	0	0	0	0

*outfitting unit

[+]converted from hours to days

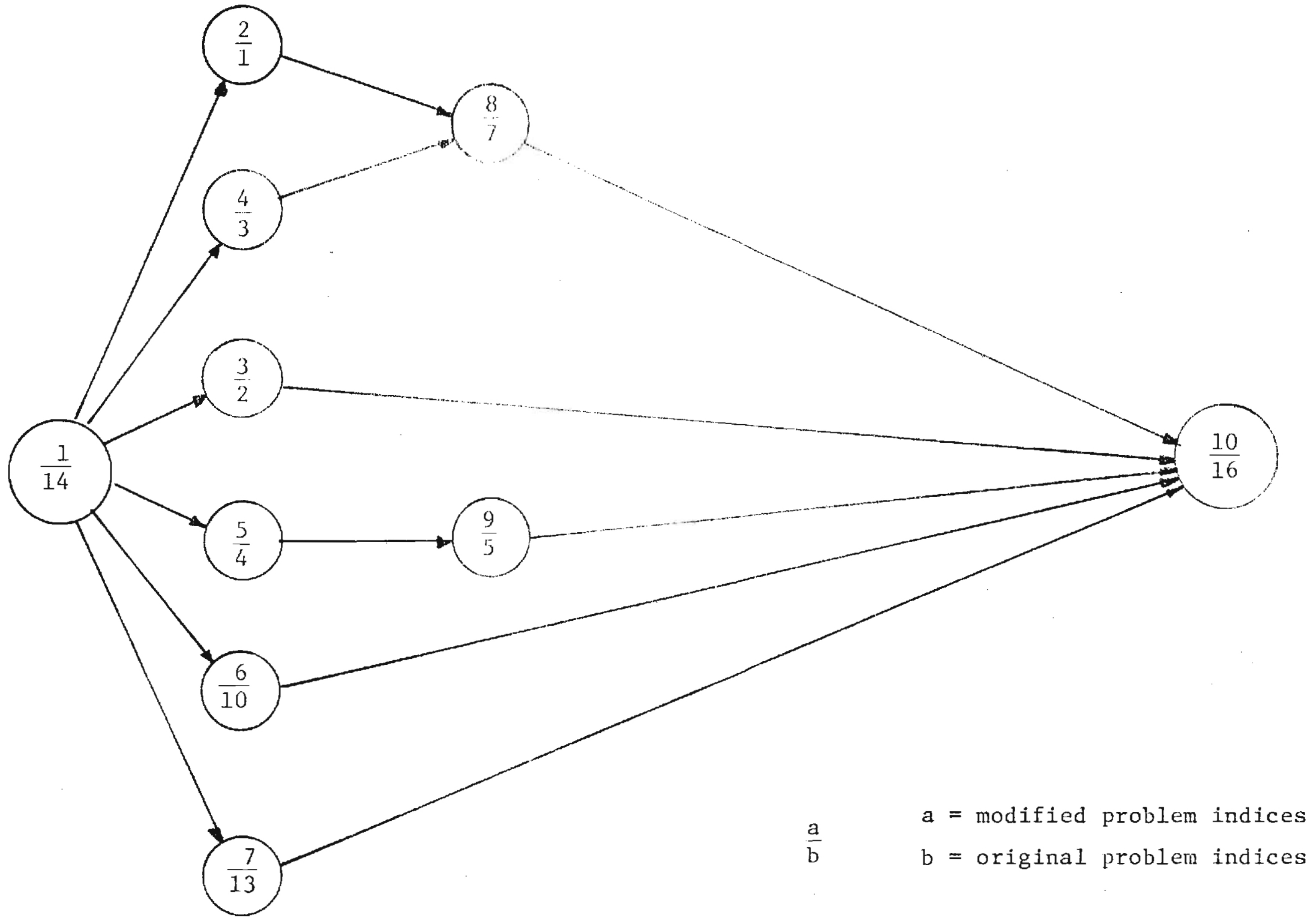

Figure 6.2: Sample Random Problem - Modified

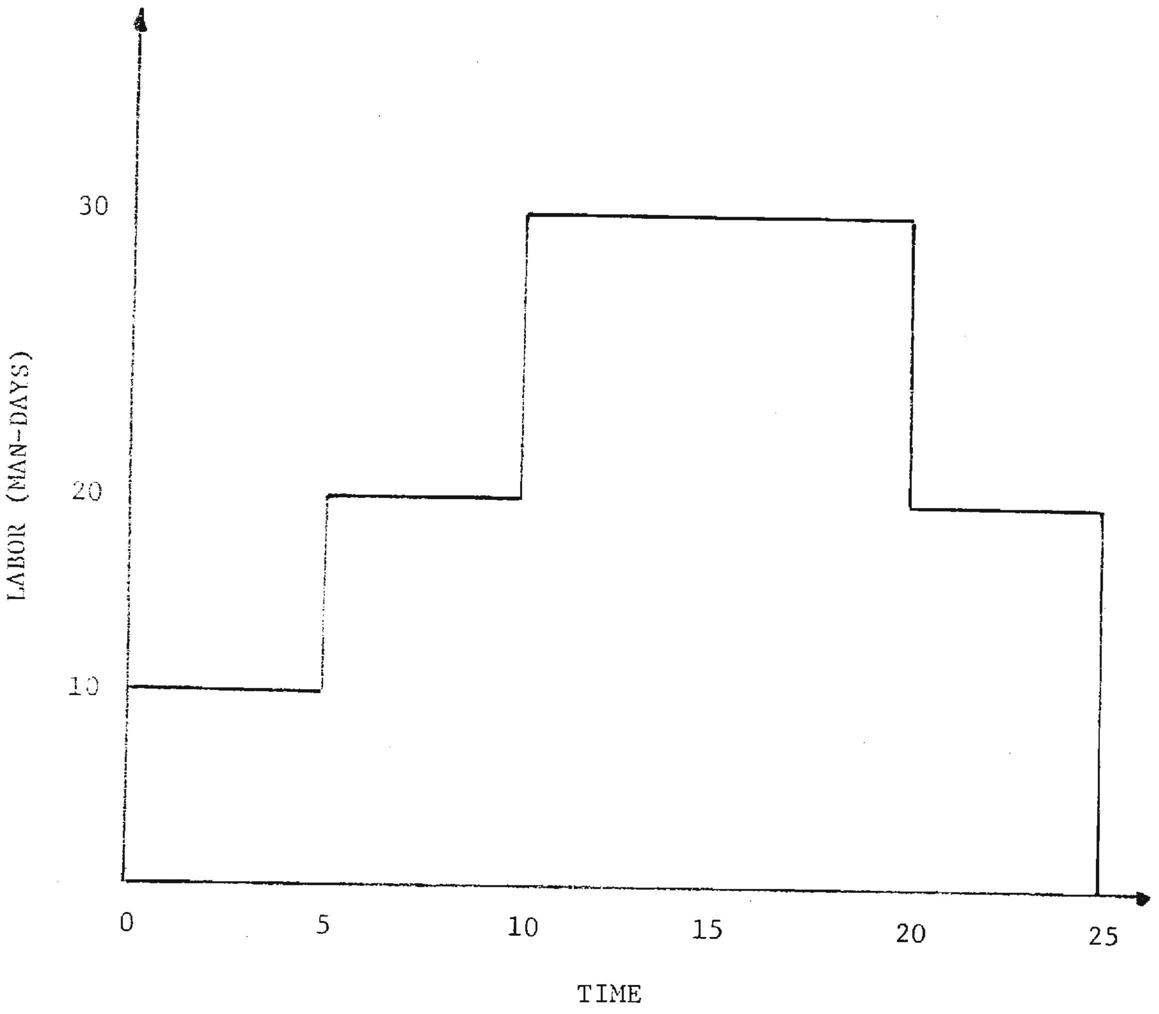

Figure 6.3: Sample Problem Resource Profile

Table 6.4: Schedule Summary for Sample Problem

```
              SOLUTION SUMMARY
              ----------------
PROFILE DURATION.................    25
MANDAYS AVAILABLE........       550.00
MANDAYS USED.............       524.00
LABOR UTILAZATION FACTOR...... .9527
VALUE SELECTED...........       65880.
VALUE SCHEDULED..........       55658.
SCHEDULE PERFORMANCE FACTOR.... .8448
NUMBER OF DEFERRED ACTIVITIES...     2
FRACTION OF DEFERRED ACTIVITIES..2000
-------------------------------------
NUMBER OF SCHEDULING DECISIONS..     6
NUMBER OF KNAPSACK PROBLEMS.....     5
KNAPSACK TIME...............       .01
TOTAL TIME..................       .06
```

6.2 TEST PROBLEMS

A set of six test problems were created using the test problem generator described in Chapter five and listed in Appendix C. The default parameters for the generator were used, resulting in the problem parameters given in Table 6.5. Each pair of problems, (1, 2), (3, 4), and (5, 6), corresponds, respectively to 60, 90 and 150 outfitting activities or work packages. This variation in problem size will provide insight into the computational requirements of the solution procedure.

6.3 TEST RESULTS

The first phase of the solution procedure is to use program SELECT to analyze the value vs. resource usage tradeoff. In the six test problems, this analysis was performed by specifying several values for outfitting window to obtain the associated tables (see Table 6.2). These tables were then examined to determine values of window duration and λ that would result in approximately 70% of the labor and 70% of the value being selected.

This initial analysis was, by far, the most computationally expensive step in the process. The total solution time for this phase depends on the number of tables generated, i.e., the number of values for outfitting window. For the 60, 90, and 150 activity problems, the solution time per table was roughly, 7 seconds, 17 seconds, and 45 seconds, respectively. These solution times, while not exorbitant, are substantially larger than the times for other steps in the analysis.

The second phase of the solution procedure is to use the selected values of outfitting window and λ to extract a reduced problem for scheduling. Table 6.6 summarizes the parameters for the extracted problems. The

Table 6.5: Test Problem Parameters

Problem Number	Number Elements	Number Units	Total Labor	Total Savings	Minimum CPM	Maximum CPM
1	40	20	4680	430773	46	70
2	50	10	4419	415433	46	74
3	70	20	6722	636485	52	84
4	75	15	6618	640402	54	86
5	120	30	10987	1122744	60	96
6	125	25	10929	1065056	56	92

Table 6.6: Extracted Problem Parameters

Problem Number	Window Length	Lambda	Number Activities	Number Precedences	Labor	Savings
1	35	81	45	70	3315	335081
2	35	75	44	65	3000	308801
3	40	80	70	104	5009	514011
4	40	80	70	103	4870	522734
5	40	85	107	167	7593	860530
6	40	75	110	169	7536	832106

time to extract the problem varies with problem size, and was roughly 2 seconds for the largest problems.

The third phase of solution procedure is to construct a resource feasible schedule, using program SCHEDULE. This requires the specification of a resource profile. For each problem, the resource profile used in these tests contained four intervals of strictly positive resource availability. The third interval was always the longest, had the largest resource level, and generally accounted for about sixty-five percent of the total labor availability.

Table 6.7 summarizes the solution statistics for the six test problems. For problems 2, 4, 5, and 6, two profiles were used, in order to see how the solution procedure behaved under varying conditions. In these instances, the second profile had the same general form, but a considerably greater duration and correspondingly smaller maximum resource level.

As discussed earlier, evaluating the quality of these solutions is quite difficult. It can be observed, however, that the labor utilization factors (total available labor ÷ labor required by scheduled activities) and the schedule performance factors (savings available ÷ savings for activities scheduled) are at 90% or better. This indicates a quite good utilization of the available resources. The percent of activities deferred to on-board outfitting is low, less than 5% for the large problems.

With regard to solution time, the method appears to perform quite well, with the largest time being 14.76 seconds. A closer examination of the execution times reveals that the time required for solving the knapsack problems in the scheduling decision rule is the largest single component. Excluding the knapsack time, the rest of the solution procedure required only 3 seconds for problems 5 and 6.

Table 6.7: Test Problem Results

Problem Number	Profile Duration	Labor Avail.	Labor Util. Factor	Sched. Perf. Factor	Fraction Deferred Activities	Solution Time (sec)
1	50	3315	.894	.898	.111	0.25
2	50	3025	.892	.906	.114	0.60
2	70	3000	.933	.933	.091	0.69
3	60	5100	.951	.974	.043	1.66
4	60	4950	.929	.925	.071	1.81
4	90	4875	.957	.942	.057	2.29
5	90	7600	.959	.959	.047	5.08
5	120	7540	.972	.963	.047	14.76
6	90	7600	.956	.963	.046	10.26
6	120	7540	.963	.963	.046	6.65

6.4 DISCUSSION

For the small sample of problems solved, it seems reasonable to conclude that the methodology has performed well with regard both to solution time and quality of solution. There is no obvious reason to suppose that the methodology would fail to perform equally well on similar problems.

At this point, the most significant undesirable aspect of the method is the substantial computing time required in the initial problem analysis. In this regard, several important points may be stated. First, the procedure as coded allowed no provision for terminating the analysis prematurely, e.g., when λ becomes large enough so that the labor selected falls below, say, 50% of the total. Since the bulk of the computing time is spent solving network flow problems, and one must be solved for each trial value of λ, this premature termination could dramatically reduce the computing time.

Second, in the results reported here, the network problem was solved "from scratch" each time, rather than as a parametric analysis of the previous solution. The reason for this was one of expediency - there were difficulties with obtaining the results using this feature of the RNET package. This is one area of the procedure that will be looked at further in the future.

Finally, it must be noted that even though this preliminary analysis appears to be expensive in terms of computing time, it also provides a wealth of information not readily available otherwise. For example, the tables generated could be combined to provide a very detailed description of the probable value associated with any level of total resource allocation. Such a table could also be obtained (with greater accuracy) by solving the scheduling problem for a large number of resource profiles. However, the latter approach would generally require more solution time if the number of allocations is more than eight or ten.

7.0 CONCLUSIONS AND RECOMMENDATIONS

This report details the following developments:

(1) a modified model of the outfit planning problem;

(2) a methodology for solving the outfit planning problem;

(3) design and implementation of experimental computer codes for solving the outfit planning problem;

(4) a computer code for generating realistic outfit planning problems conforming to the modified model; and

(5) experimental evaluation of the methodology.

The modified model of the outfit planning problem may or may not prove to be useful in practice - only time can provide that evaluation. However, should the model prove to be a reasonable one, the methodology that has been developed in this research provides an effective and efficient solution procedure.

This initial research provides a basis for further research along two fronts. First, with regard to practical shipyard planning, it would seem reasonable to consider some empirical studies to determine the applicability of this model or similar models, and, if appropriate, to attempt an in situ evaluation.

This research also provides a jumping-off point for a wide range of methodological studies of large scale planning and scheduling problems. As indicated earlier, there are a number of refinements and extensions to the methods that were developed and tested. Moreover, these methods also appear to have promise for other related problems, such as the classical resource constrained project scheduling problem.

REFERENCES

1. A Manual on Planning and Production Control for Shipyard Use, Bath Iron Works, Bath, Maine.

2. Andrews, V. B., et al., "Conceptual Design of a Mechanized Shipyard for Fast Deployment Logistics (X) Production," National Technical Information Service, U.S. Department of Commerce, AD-752-595, December, 1965, pp. 1-88.

3. Balas, E., and E. Zemel, "Solving Large Zero-One Knapsack Problems," Management Sciences Research Report No. 408(R), Graduate School of Industrial Administration, Carnegie-Mellon University, Pittsburgh, PA, 1977.

4. Battersby, A., Network Analysis for Planning and Scheduling, St. Martin's Press, New York, 1970.

5. Bennington, G. E., and L. F. McGinnis, "A Critique of Project Planning with Constrained Resources," Proceedings of the Symposium on Scheduling Theory and Its Applications, S. E. Elmaghraby, ed., Springer-Verlag, 1973.

6. Bradley, G. H., G. G. Brown, and G. W. Graves, "Design and Implementation of Large Scale Primal Transshipment Algorithms," Management Science, Vol. 24, No. 1, 1977.

7. Chirillo, L. D., presentation to Chesapeake Section, SNAME, May, 1979.

8. Chirillo, L. D., private correspondence, Todd Pacific Shipyards Corporation, Seattle Division, January, 1979.

9. Chirillo, L. D., and C. S. Jonson, Outfit Planning Manual, National Shipbuilding Research Program, Project SP-IV-D, 1979.

10. Cooper, D. F., "Heuristics for Scheduling Resource-Constrained Projects: An Experimental Investigation," Management Science, Vol. 22, No. 11, 1976.

11. Davis, E. W., "Resource Allocation in Project Network Models - A Survey," Journal of Industrial Engineering, April, 1966.

12. Davis, E. W., "Project Scheduling under Resource Constraints - Historical Review and Categorization of Procedures," AIIE Transactions, Vol. 5, No. 4, 1973.

13. Dewitte, L., "Manpower Leveling of PERT Networks," Data Processing for Science/Engineering, Vol. 2, No. 2, 1964.

14. Elmaghraby, S. E., "On the Expected Duration of PERT Type Networks," Management Science, Vol. 13, No. 5, 1967.

15. Elmaghraby, S. E., Some Network Models in Management Science, Springer-Verlag, 1970.

16. Elmaghraby, S. E., Activity Networks: Project Planning and Control by Network Models, Wiley, 1977.

17. Fox, B. L., "Data Structures and Computer Science Techniques in Operations Research," Operations Research, Vol. 26, No. 5, 1978.

18. Fulkerson, D. R., "A Network Flow Computation for Project Cost Curves," Management Science, Vol. 7, No. 2, 1961.

19. Garvey, J. J., "The National Shipbuilding Research Program 1971-1976," presented to the Philadelphia Section Society of Naval Architects and Marine Engineers, April, 1976.

20. Geoffrion, A. M., "Lagrangian Relaxation for Integer Programming," Mathematical Programming Study 2: Approaches to Integer Programming, 1975.

21. Glover, F., D. Karney, and D. Klingman, "Implementation and Computational Comparisons of Primal, Dual, and Primal-Dual Computer Codes for Minimum Cost Network Flow Problems," Networks, Vol. 4, No. 3, 1974.

22. Goldbach, R. A., "Application of Preoutfitting During Construction of Ammunition Ships AE 32-35," Marine Technology, January, 1973.

23. Graves, G., "Multicommodity Distribution Systems Design by Benders' Decomposition," Management Science, Vol. 20, 1974.

24. Graves, R. J., and L. F. McGinnis, "A Decision Support System for the Pre-Outfitting/Outfitting Problem," Department of Commerce Contract No. DO-A01-78-00-3074.

25. Graves, R. G., L. F. McGinnis, and L. D. Bailey, "The Outfit Planning Problem," Department of Commerce Contract No. DO-A01-78-00-3074, June 15, 1979.

26. Grigoriadis, M. D., and T. Hsu, "RNET: The Rutgers Minimum Cost Network Flow Subroutines," Department of Computer Science, Rutgers University - Busch Campus, New Brunswick, New Jersey 08903, October, 1979.

27. Gross, D., and C. E. Pinkus, "Optimal Allocation of Ships to Yards for Regular Overhauls," Technical Memorandum, Serial TM-63095, Office of Naval Research, Project NR 347-020, May, 1972.

28. Grubbs, F. E., "Attempts to Validate PERT Statistics or 'Picking on PERT,'" Operations Research, Vol. 10, No. 6, 1962, pp. 912-915.

29. Helgason, R. V., and J. L. Kennington, "NETFO Program Documentation," Technical Report IEOR 76011, Department of Industrial Engineering and Operations Research, Southern Methodist University, September, 1976.

30. Hurst, R., "Some Production Research Activities on Steelwork and Outfitting," The Society of Naval Architects and Marine Engineers, June, 1968.

31. J. J. Henry Co., Inc., "Modular Design Applications Study (Deckhouse and Outfit)," The U.S. Department of Commerce Maritime Administration, Contract No. MA-4358, PB 178196, 1967, pp. 1-57.

32. Jolliff, J. V., CDR, USN, "Modular Ship Design Concepts," Naval Engineers Journal, Vol. 86, No. 5, October, 1974.

33. Johnson, C., "Scheduling, Planning and Reporting Data Information Systems," National Steel and Shipbuilding Company, San Diego, Calif.

34. Karp, R., "On the Computational Complexity of Combinatorial Problems," Networks, Vol. 5, 1975.

35. Kelley, J., and M. Walker, "Critical-Path Planning and Scheduling," Proceedings of the Eastern Joint Computer Conference, 1959.

36. Kelley, J. E., Jr., "Critical-Path Planning and Scheduling: Mathematical Basis," Operations Research, Vol. 9, pp. 296-320, 1961.

37. Kelley, J. E., "The Critical Path Method: Resources Planning and Scheduling," Industrial Scheduling, J. R. Math and G. L. Thompson, eds., Prentice-Hall, 1963.

38. Levy, F. K., G. L. Thompson, and J. D. Wiest, "Mathematical Basis of the Critical Path Method," Industrial Scheduling, J. F. Muth and G. L. Thompson, eds., Prentice-Hall, 1963.

39. Levy, F. K., G. L. Thompson, and J. D. Wiest, "Multi-Ship, Multi-Shop Workload Smoothing Program," Naval Research Logistics Quarterly, Vol. 9, No. 1, 1963.

40. Malcomb, D. G., J. H. Roseboom, C. E. Clark, and W. Fazar, "Application of a Technique for Research and Development Program Evaluation," Operations Research, Vol. 7, pp. 646-669, 1959.

41. Marsten, R. E., "Integer Programming Algorithms: A Framework and State-of-the-Art Survey," Management Science, Vol. 18, No. 7, 1972.

42. McBride, R., "Lagrangian Relaxation Applied to Capacitated Facility Location Problems," AIIE Transactions, Vol. 10, No. 1, 1978.

43. McGinnis, L. F., and R. J. Graves, "A Mathematical Model for the Outfit Planning Problem," Department of Commerce Contract No. DO-A01-78-00-3074, October 31, 1979.

44. McGinnis, L. F., and H. L. W. Nuttle, "The Project Coordinator's Problem," OMEGA, Vol. 6, No. 4, 1978.

45. Moder, J. J., and C. R. Phillips, Project Management with CPM and PERT, Reinhold, 1964.

46. Modular Design Applications Study (Deckhouse and Outfit), The U.S. Department of Commerce Maritime Administration, Contract No. MA-4358, PB 178196, 1967, pp. 1-57.

47. Naus, R. M., "An Efficient Algorithm for the 0-1 Knapsack Problem," Management Science, Vol. 23, 1976.

48. Naus, R. M., "The 0-1 Knapsack Problem with Multiple Choice Constraints," University of Missouri - St. Louis, 1976.

49. Padberg, M., and T. Shaftel, "The Modular Design Problem: Convexity of the Constraint Set," Management Science Research Report No. 245, Carnegie-Mellon University, March, 1971.

50. Potts, W. R., and W. L. Cuthbert, "L.N.G. Carriers Using the Conch Containment System," The Society of Naval Architects and Marine Engineers, No. 11A, 1975, pp. 21-40.

51. Proceedings for Shipbuilding Industrial/Production Engineering Workshop, R. J. Graves, ed., U.S. Department of Commerce, February, 1978.

52. Sahni, S., and E. Horowitz, "Combinatorial Problems: Reducibility and Approximation," Operations Research, Vol. 26, No. 5, 1978.

53. Scheduling, Planning and Reporting Data Information System, National Steel and Shipbuilding Company, San Diego, California.

54. Shaftel, T. L., Thompson, G. L., "A Simplex-Like Algorithm for the Continuous Modular Design Problem," Management Science Research Report No. 248, Carnegie-Mellon University, under contract N00014-67-A-0314-0007 NR Q47-048 U.S. Navy Office of Naval Research, January, 1972.

55. Shaftel, T. L., Thompson, G. L., "The Continuous Multiple Modular Design Problem," Management Science Research Report No. 254, Carnegie-Mellon University, under contract N00014-67-A-0314-0007 NR 047-048 with the U.S. Office of Naval Research, January, 1972.

56. Stinson, J. P., E. W. Davis, and B. M. Khumawala, "Multiple Resource-Constrained Scheduling Using Branch and Bound," AIIE Transactions, Vol. 10, No. 3, 1978.

57. Talbot, F. B., and J. H. Patterson, "An Efficient Integer Programming Algorithm with Network Cuts for Solving Resource-Constrained Scheduling Problems," Management Science, Vol. 24, No. 11, 1978.

58. Wiest, J. D., "A Heuristic Model for Scheduling Large Projects with Limited Resources," Management Science, February, 1967.

APPENDIX A

PROGRAM SELECT LISTING

```
      PROGRAM SELECT(DATA,LTABLE,INPUT,OUTPUT,TAPE1=DATA,
     +  TAPE5=INPUT,
     *                 TAPE6=OUTPUT,TAPE9=LTABLE)
C
C-----THIS PROGRAM PERFORMS TIME SCREENING FOR A SPECIFIED
C        WINDOW
C-----LENGTH, *TEND*, AND THEN GENERATES THE LOCUS OF
C        OPTIMAL VALUE
C-----VS. RESOURCE POINTS, IN THE FILE *LTABLE*.  USING THE
C        LATTER,
C-----A SPECIFIC RESOURCE LEVEL (AND CORRESPONDING
C        MULTIPLIER, LAMBDA)
C-----CAN BE SELECTED.  THIS, ALONG WITH PROBLEM DATA IS
C        THEN INPUT TO
C-----PROGRAM *SCHEDUL*, WHICH DETERMINES A RESOURCE
C        FEASIBLE SCHEDULE.
C
C-----RNET COMMON BLOCK**RNET COMMON BLOCK**RNET COMMON
C        BLOCK**
C
C-----NACT-LONG ARRAYS ARE
      INTEGER P(250),F(250),D(250),U(250),ARC(250),X(250),
     +  R(250)
C-----NARCS-LONG ARRAYS ARE
      INTEGER FROM(1000),TO(1000),C(1000),H(1000)
C-----CANDIDATE LIST ARRAY IS
      INTEGER PCAND(250)
C-----OTHER ARRAYS ARE
      INTEGER RTN(20),LPR(12)
      DIMENSION RTT(2)
C-----PARAMETERS ARE
      INTEGER BT,PT,PAS,MW,IRT,MXIT,NCAND
      REAL FRQ,P0,PBAR,CP
C-----RNET NAMED COMMONS ARE
      COMMON /A/FROM/B/TO/C/C/D/H/E/P/F/F/G/D/H/U/I/ARC/J/X/
     +  K/R/N/PCAND
      COMMON /L/BT,MW,PT,NNODES,NARCS,IRT,MXIT,FRQ,P0,PBAR,
     +  CP,PAS,RTN,
     *              LPR,RTT,NCAND
C
C-----RNET COMMON BLOCK**RNET COMMON BLOCK**RNET COMMON
C        BLOCK**
C
C
C-----PROBLEM COMMON BLOCK**PROBLEM COMMON BLOCK**PROBLEM
C        COMMON BLOCK**
C
      INTEGER PPTR(250),SPTR(250),TL(250),DMIN(250),
     +  DMAX(250),SAV(250),
     *        NTOP(250),PRED(1000),SUCC(1000)
```

```
       COMMON/PROB/PPTR,PRED,SPTR,SUCC,TL,DMIN,DMAX,SAV,NTOP,
      *  NACT,
      *              MAXACT,MAXARC
C
C-----PROBLEM COMMON BLOCK**PROBLEM COMMON BLOCK**PROBLEM
C        COMMON BLOCK**
C
C
C-----SEARCH COMMON BLOCK**SEARCH COMMON BLOCK**SEARCH
C        COMMON BLOCK**
C
       REAL  POINTS(3000),INTVLS(6000),RHSDEL(250),PNTREC(3),
      *  INTREC(6)
       COMMON/SEARCH/POINTS,INTVLS,RHSDEL,MAXPTS,MAXIVL,SCAL,
      *  TLAB,TSAV,
      *                PNTREC,INTREC
C
C-----SEARCH COMMON BLOCK**SEARCH COMMON BLOCK**SEARCH
C        COMMON BLOCK**
C
C
C-----ADDITIONAL DECLARATIONS SPECIFIC TO THIS PROGRAM
       INTEGER ES(250),UNIT1,UNIT2,UNIT3,UNIT4,SIZEGO,SIZENG,
      *  TEND
       REAL  LLAM,LRES,LVAL
       COMMON/PARMS/UNIT1,UNIT2,UNIT3,UNIT4
       DATA  MAXACT,MAXARC/247,1000/,ITHREE,ISIX/3,6/
       DATA  UNIT1,UNIT2,UNIT3,UNIT4/1,9,5,6/
       DATA  MAXPTS,MAXIVL/3000,6000/,EPS/.0001/,SCAL/10000./
       DATA  BT,PT,INT,MXIT,FRQ,PC,PBAR,NOAND/48,0,1,500,8.,
      *  1.,1.5,0/
C
C
C-----BEGIN PROGRAM *SELECT*//BEGIN PROGRAM *SELECT*//
C
C-----GET PROBLEM DATA FROM UNIT1
       WRITE(UNIT4,10004)
10004  FORMAT(" ENTER VALUES FOR SCALE,EPS,IPRNT")
       READ(UNIT3,*)SCAL,EPS,IPRNT
       CALL  DATIN(ES,UNIT1,UNIT4,IPRNT)
C-----GET WINDOW DURATION FROM UNIT2 (THIS CAN BE A
C-----IN AN INTERACTIVE SESSION, OR A DATA FILE IN BATCH
C        MODE)
1      WRITE(UNIT4,10000)
10000  FORMAT(" WHAT VALUE FOR TEND")
       READ(UNIT3,*)TEND
       IF(TEND.EQ.0)GO TO 9999
C
C-----PROBLEM DATA INPUT COMPLETE.  NOW GET TOPOLOGICAL
C        ORDERING
```

```
      CALL TSORT(IRTN,TO)
      IF(IRTN.NE.1)CALL ERRSTP(IRTN,UNIT4)
      IF(IPRNT.GT.1)
     *CALL ECHO1(ES,UNIT4)
C
C-----TIME SCREENING BEGINS HERE.  CALCULATE CPM EARLY
C-----SCHEDULE, USING DMIN FOR LARGEST CANDIDATE POOL.
      ITIME=0
      NODE=1
      CALL EARLY1(ITIME,NODE,DMIN,ES)
      IF(IPRNT.GT.1)
     *CALL ECHO2(ES,UNIT4)
C-----REORDER THE ACTIVITIES SO THAT ALL THE DEFERRED
C       ACTIVITIES
C-----APPEAR AFTER THE SELECTED ACTIVITIES IN *NTOP*.  USE
C       *C* AND
C-----*H* AS THE LISTS TO RECEIVE THE SPLIT *NTOP* LIST.
      SIZEGO=0
      SIZENG=0
      DO 100 II=1,NACT
        I=NTOP(II)
        IF(ES(I)+DMIN(I).LE.TEND)THEN
          SIZEGO=SIZEGO+1
          C(SIZEGO)=I
        ELSE
          SIZENG=SIZENG+1
          H(SIZENG)=I
        ENDIF
100   CONTINUE
      DO 200 I=1,SIZEGO
        NTOP(I)=C(I)
200   CONTINUE
      DO 300 I=1,SIZENG
        NTOP(SIZEGO+I)=H(I)
300   CONTINUE
      IF(IPRNT.GT.0)
     *WRITE(UNIT4,90000)SIZEGO
90000 FORMAT(" SIZEGO=",I5)
      IF(IPRNT.GT.2)
     *CALL ECHO1(ES,UNIT4)
C-----NOTE THAT *ES* IS NO LONGER IN CORRECT ORDER, SO DO
C-----NOT USE IT WITHOUT FIRST CALLING SUBROUTINE EARLY.
C
C-----TIME SCREENING COMPLETE.  READY TO SET UP FLOW
C       NETWORK DATA.
C-----USE ONLY THE FIRST *SIZEGO* ACTIVITIES IN NTOP FOR
C       NETWORK.
      IF(SIZEGO.LE.1)GO TO 9999
      NNODES=SIZEGO+1
C-----NODE *NNODES* IS THE SUPERSOURCE/SINK FOR W(J) AND
C       S(J) FLOWS.
```

```
C
C-----CREATE A MAPPING FROM ORIGINAL ACTIVITIES INTO THE
C       NTOP
C-----SEQUENCE USING *P* .  THIS MAPPING CAN BE DISCARDED
C-----SOON AS THE NETWORK STRUCTURE IS COMPLETE.
      DO 400 I=1,NACT
        P(NTOP(I))=I
400   CONTINUE
C
C-----IF P(J).GT.SIZEGO THEN J HAS BEEN DEFERRED.
C-----SET UP RNET ARC ARRAYS AND NODE FLOWS FOR LAMBDA=0
      NARCS=0
      TSAV=0.
      TLAB=0.
      DO 500 I=1,SIZEGO
        NARCS=NARCS+1
        FROM(NARCS)=NNODES
        TO(NARCS)=I
        C(NARCS)=1
        H(NARCS)=0
        NARCS=NARCS+1
        FROM(NARCS)=I
        TO(NARCS)=NNODES
        C(NARCS)=0
        H(NARCS)=0
        IFIRST=SPTR(NTOP(I))
        ILAST=SPTR(NTOP(I)+1)-1
        IF(IFIRST.LE.ILAST)THEN
          DO 420 II=IFIRST,ILAST
            IF(P(SUCC(II)).GT.SIZEGO)GO TO 420
            NARCS=NARCS+1
            FROM(NARCS)=I
            TO(NARCS)=P(SUCC(II))
            C(NARCS)=0
            H(NARCS)=0
 420      CONTINUE
        ENDIF
        X(I)=-SAV(NTOP(I))*SCAL
        TSAV=TSAV+X(I)
        TLAB=TLAB+TL(NTOP(I))
500   CONTINUE
      X(NNODES)=-TSAV
      IF(IPRNT.GT.2)THEN
      WRITE(UNIT4,90001)
90001 FORMAT(" RNET DATA")
      DO 510 II=1,NARCS
        WRITE(UNIT4,90002)FROM(II),TO(II),C(II),H(II)
510   CONTINUE
90002 FORMAT(" ",4I10)
      DO 520 II=1,NNODES
```

```
          WRITE(UNIT4,90002)II,X(II)
520     CONTINUE
        ENDIF
C-----RNET ARRAYS INITIALIZED.   READY TO START SEARCH OVER
C-----LAMBDA.  FIRST RESET *P* ARRAY TO ZERO
        DO 600 I=1,NACT
          P(I)=0
600     CONTINUE
C
C-----INITIALIZE FOR SEARCH PHASE
        NPNT=0
        NINT=0
        RLAM=0.
        LLAM=0.
        DO 1000 I=1,SIZEGO
          IF(TL(NTOP(I)).EQ.0)GO TO 1000
          RLAM=MAX(RLAM,(FLOAT(SAV(NTOP(I)))/TL(NTOP(I))))
1000    CONTINUE
90003 FORMAT(/" INIT RLAM= ",F15.12)
        CALL  SOLVE(LLAM,LLAM,LRES,LVAL,IPRNT)
        RRES=0.
        RVAL=0.
        PNTREC(1)=RLAM
        PNTREC(2)=RRES
        PNTREC(3)=RVAL
        CALL  PUSH(POINTS,MAXPTS,NPNT,ITHREE,PNTREC)
        IF(IPRNT.GT.3)
       *WRITE(UNIT4,90004)NPNT,PNTREC
90004 FORMAT(/" PUSH POINT ",I5,3E15.8)
        PNTREC(1)=LLAM
        PNTREC(2)=LRES
        PNTREC(3)=LVAL
        CALL  PUSH(POINTS,MAXPTS,NPNT,ITHREE,PNTREC)
        IF(IPRNT.GT.3)
       *WRITE(UNIT4,90004)NPNT,PNTREC
        RLAM=RLAM-EPS
        CALL  SOLVE(RLAM,0.,RRES,RVAL,IPRNT)
        INTREC(1)=LLAM
        INTREC(2)=RLAM
        INTREC(3)=LRES
        INTREC(4)=RRES
        INTREC(5)=LVAL
        INTREC(6)=RVAL
        CALL  PUSH(INTVLS,MAXIVL,NINT,ISIX,INTREC)
        IF(IPRNT.GT.3)
       *WRITE(UNIT4,90005)NINT,INTREC
90005 FORMAT(/" PUSH INTERVAL ",I5,6E15.8)
        OLAM=RLAM
C-----INITIALIZATION COMPLETE.   BEGIN SEARCH PHASE.
2000    IF(NINT.EQ.0)GO TO 3000
```

```
      CALL POP(INTVLS,MAXIVL,NINT,ISIX,INTREC)
      IF(IPRNT.GT.3)
     + WRITE(UNIT4,90006)NINT+1,INTREC
90006 FORMAT(/" POP INTERVAL ",I5,6E15.8)
      LLAM=INTREC(1)
      RLAM=INTREC(2)
      LRES=INTREC(3)
      RRES=INTREC(4)
      LVAL=INTREC(5)
      RVAL=INTREC(6)
      PLAM=PROJECT(LLAM,RLAM,LRES,RRES,LVAL,RVAL)
      PLAM=PLAM+EPS
C-----TO GET RIGHT HAND DERIVATIVE AT PLAM
      IF(IPRNT.GT.3)
     * WRITE(UNIT4,90007)LLAM,RLAM,PLAM
90007 FORMAT(/" PROJECT ",2E15.8," TO ",E15.8)
      CALL SOLVE(PLAM,OLAM,PRES,PVAL,IPRNT)
2100  IF(PRES.EQ.RRES.OR.PRES.EQ.LRES)GO TO 2500
C     NOT A BREAK POINT  DIVIDE THE INTERVAL
        INTREC(1)=PLAM
        INTREC(2)=RLAM
        INTREC(3)=PRES
        INTREC(4)=RRES
        INTREC(5)=PVAL
        INTREC(6)=RVAL
        CALL PUSH(INTVLS,MAXIVL,NINT,ISIX,INTREC)
        IF(IPRNT.GT.3)
     *  WRITE(UNIT4,90005)NINT,INTREC
        RLAM=PLAM
        RRES=PRES
        RVAL=PVAL
        PLAM=PROJECT(LLAM,PLAM,LRES,PRES,LVAL,PVAL)
        PLAM=PLAM+EPS
        IF(IPRNT.GT.3)
     *  WRITE(UNIT4,90007)LLAM,RLAM,PLAM
        CALL SOLVE(PLAM,OLAM,PRES,PVAL,IPRNT)
        GO TO 2100
2500  CONTINUE
C-----FOUND ANOTHER BREAK POINT
      PNTREC(1)=PLAM
      PNTREC(2)=PRES
      PNTREC(3)=PVAL
      CALL PUSH(POINTS,MAXPTS,NPNT,ITHREE,PNTREC)
      IF(IPRNT.GT.3)
     * WRITE(UNIT4,90004)NPNT,PNTREC
      GO TO 2000
3000 CONTINUE
C-----SEARCH PHASE COMPLETE.  *POINTS* CONTAINS THE
C     SEQUENCE OF
C-----BREAK POINTS.  WRITE OUT THE INFO IN FILE *LTABLE*
C     AND STOP.
```

```
C
      WRITE(UNIT2,10001)TEND,TEND
10001 FORMAT("1",6X,"BREAKPOINTS OF V(",I3,",LAMBDA) AND
     *   G(LAMBDA)"//
     *7X,"LAMBDA",10X,"V(",I3,",LAMBDA)",10X,"G(LAMBDA)",
     *   10X,"VALUE"/
     *7X,6("-"),10X,13("-"),10X,9("-"),10X,5("-")//)
      DO 4000 I=2,NPNT
        II=(I-1)*3
        VAL=POINTS(II+1)*POINTS(II+2)+POINTS(II+3)
        WRITE(UNIT2,10002)POINTS(II+1),POINTS(II+3),
     *   POINTS(II+2),VAL
4000  CONTINUE
      IF(POINTS(2).NE.POINTS((NPNT-1)*3+2))THEN
        VAL=POINTS(1)*POINTS(2)+POINTS(3)
        WRITE(UNIT2,10002)POINTS(1),POINTS(3),POINTS(2),VAL
      ENDIF
10002 FORMAT(2X,E12.6,9X,E12.6,9X,E12.6,5X,E12.6)
      GO TO 1
9999  WRITE(UNIT+,10003)
10003 FORMAT("1"//" *****RUN TERMINATED NORMALLY****")
      STOP
      END
      SUBROUTINE SOLVE(PLAM,OLAM,PRES,PVAL,IPRNT)
C
C-----TO SET UP AND SOLVE THE DUAL NETWORK FLOW PROBLEM FOR
C        A GIVEN
C-----LAMBDA.   THE METHOD IS TO CONTINUE FROM THE SOLUTION
C        FOR THE
C-----PREVIOUS VALUE OF LAMBDA, USING THE RHS
C        PARAMETERIZATION FACILITY
C-----PROVIDED IN RNET.
C
C
C-----RNET COMMON BLOCK**RNET COMMON BLOCK**RNET COMMON
C        BLOCK**
C
C-----NACT-LONG ARRAYS ARE
      INTEGER P(250),F(250),D(250),U(250),ARC(250),X(250),
     *   R(250)
C-----NARCS-LONG ARRAYS ARE
      INTEGER FROM(1000),TO(1000),C(1000),H(1000)
C-----CANDIDATE LIST ARRAY IS
      INTEGER PCAND(250)
C-----OTHER ARRAYS ARE
      INTEGER RTN(20),LPR(12)
      DIMENSION RTT(2)
C-----PARAMETERS ARE
      INTEGER BT,PT,PAS,MW,IRT,MXIT,NCAND
      REAL FRQ,PC,PBAR,CP
C-----RNET NAMED COMMONS ARE
      COMMON /A/FROM/B/TO/C/C/D/H/E/P/F/F/G/D/H/U/I/ARC/J/X/
     *   K/R/N/PCAND
      COMMON /L/BT,MW,PT,NNODES,NARCS,IRT,MXIT,FRQ,PC,PBAR,
     *   CP,PAS,RTN,
     *            LPR,RTT,NCAND
C
C-----RNET COMMON BLOCK**RNET COMMON BLOCK**RNET COMMON
C        BLOCK**
C
C
C-----PROBLEM COMMON BLOCK**PROBLEM COMMON BLOCK**PROBLEM
C        COMMON BLOCK**
C
      INTEGER PPTR(250),SPTR(250),TL(250),DMIN(250),
     *   DMAX(250),SAV(250),
     *         NTOP(250),PRED(1000),SUCC(1000)
      COMMON/PROB/PPTR,PRED,SPTR,SUCC,TL,DMIN,DMAX,SAV,NTOP,
     *   NACT,
     *              MAXACT,MAXARC
C
C-----PROBLEM COMMON BLOCK**PROBLEM COMMON BLOCK**PROBLEM
C        COMMON BLOCK**
```

```
C
C
C-----SEARCH COMMON BLOCK**SEARCH COMMON BLOCK**SEARCH
C       COMMON BLOCK**
C
      REAL  POINTS(3000),INTVLS(6000),PNTREC(3),INTREC(6)
      INTEGER RHSDEL(250)
      COMMON/SEARCH/POINTS,INTVLS,RHSDEL,MAXPTS,MAXIVL,SCAL,
     *  TLAB,TSAV,
     *                PNTREC,INTREC
C
C-----SEARCH COMMON BLOCK**SEARCH COMMON BLOCK**SEARCH
C       COMMON BLOCK**
C
C
      INTEGER UNIT1,UNIT2,UNIT3,UNIT4
      COMMON/PARMS/UNIT1,UNIT2,UNIT3,UNIT4
C-----THE ARC DATA HAS ALREADY BEEN PREPARED.  IF THIS IS
C       THE FIRST
C-----CALL  (PLAM=0) THEN INITIALIZE RNET, OTHERWISE, UPDATE
C       THE RHS
C-----AND RESTART.
      IF(IPRNT.GT.3)
     *WRITE(UNIT4,90000)PLAM,OLAM
90000 FORMAT(/" SOLVE ",2F10.7)
      IRT=NNODES
      IF(PLAM.EQ.0.0)THEN
        OLAM=PLAM
        CALL RNET
      ELSE
        ITEMP=0
        DO 100 I=1,NNODES-1
          X(I)=-1*(SAV(NTOP(I))-PLAM*TL(NTOP(I)))*SCAL
          ITEMP=ITEMP+X(I)
100     CONTINUE
        X(NNODES)=-ITEMP
        IF(IPRNT.GT.3)
     *  WRITE(UNIT4,90002)(X(I),I=1,NNODES)
90002   FORMAT(/" RHS:  ",/(8I10))
        CALL RNET
        OLAM=PLAM
      ENDIF
      IF(RTN(1).NE.0)THEN
        CALL RNWFPA(UNIT4)
        CALL ERRSTP(600,UNIT4)
      ENDIF
C
C-----NOW DETERMINE SOLUTION VALUE AND RESOURCE USAGE
C
      IF(IPRNT.GT.3)
     *WRITE(UNIT4,90003) (U(I),I=1,NNODES)
90003 FORMAT(/" DUAL VARS: ",/(8I10))
      DVAL=RTN(2)/SCAL
      PVAL=0.
      PRES=0.
      DO 200 I=1,NNODES-1
        IF(U(I).NE.0)THEN
          PVAL=PVAL+SAV(NTOP(I))-PLAM*TL(NTOP(I))
          PRES=PRES+TL(NTOP(I))
        ENDIF
200   CONTINUE
      IF(IPRNT.GT.3)
     *WRITE(UNIT4,90001)DVAL,PVAL,PRES
90001 FORMAT(/" DVAL= ",E15.8," PVAL= ",E15.8," PRES= ",
     *  E15.8)
      IF(ABS(DVAL-PVAL).GT.NNODES/SCAL)THEN
        CALL RNWFPA(UNIT4)
        CALL ERRSTP(700,UNIT4)
      ENDIF
      RETURN
      END
      SUBROUTINE DATIN(SCRATCH,IN,OUT,IPRNT)
C
C
C-----PROBLEM COMMON BLOCK**PROBLEM COMMON BLOCK**PROBLEM
C       COMMON BLOCK**
C
      INTEGER PPTR(250),SPTR(250),TL(250),DMIN(250),
     *  DMAX(250),SAV(250),
     *        NTOP(250),PRED(1000),SUCC(1000)
      COMMON/PROB/PPTR,PRED,SPTR,SUCC,TL,DMIN,DMAX,SAV,NTOP,
     *  NACT,
     *            MAXACT,MAXARC
C
C-----PROBLEM COMMON BLOCK**PROBLEM COMMON BLOCK**PROBLEM
C       COMMON BLOCK**
C
C
      INTEGER SCRATCH(1),OUT,TLAB,TSAV
      DATA ITIME,NODE/0,1/
C-----READ ACTIVITY SPECS (RJG TAPE2)
C-----NOTE THAT TAPE3 IS APPENDED TO THE END OF TAPE2 FROM
C-----KAHAN'S PROBLEM GENERATOR.
      READ(IN,*)NE,NU
      NK=NE+NU
      NACT=NK+2
      IF(NACT.GT.MAXACT)CALL ERRSTP(900,OUT)
C-----ACTIVITY NK+1 WILL BE THE DUMMY START NODE
C-----ACTIVITY NK+2 WILL BE THE DUMMY END NODE
      TLAB=0.
      TSAV=0
      DO 1000 I=1,NK
        READ(IN,*)IDUM
        IF(IDUM.NE.NK+I)CALL ERRSTP(1000,OUT)
        READ(IN,*)TL(I),DMIN(I),DMAX(I),SAV(I),IDUM
        TL(I)=TL(I)/8.0
        TLAB=TLAB+TL(I)
        TSAV=TSAV+SAV(I)
1000  CONTINUE
C
C-----READ PRECEDENCE FILES (RJG TAPE2) AND ADD ARCS
C-----NODE NK+1 (IN *NTOP*) OR TO NODE KN+2 (IN
C-----*SCRATCH*) AS REQUIRED.
      NPPTR=1
      NSPTR=1
      NSCRTCH=0
      NNTOP=0
      DO 2000 I=1,NK
        SPTR(I)=NSPTR
        PPTR(I)=NPPTR
        READ(IN,*)NS,NP
```

```
          IF(NS.EQ.0)THEN
            SUCC(NSPTR)=NK+2
            NSPTR=NSPTR+1
            NSCRTCH=NSCRTCH+1
            SCRATCH(NSCRTCH)=I
          ELSE
            DO 1100 K=1,NS
              READ(IN,*)IDUM
              SUCC(NSPTR)=IDUM-NK
              NSPTR=NSPTR+1
              IF(NSPTR.GT.MAXARC)CALL ERRSTP(900,OUT)
 1100       CONTINUE
          ENDIF
          IF(NP.EQ.0)THEN
            PRED(NPPTR)=NK+1
            NPPTR=NPPTR+1
            NNTOP=NNTOP+1
            NTOP(NNTOP)=I
          ELSE
            DO 1200 K=1,NP
              READ(IN,*)IDUM
              PRED(NPPTR)=IDUM-NK
              NPPTR=NPPTR+1
              IF(NPPTR.GT.MAXARC)CALL ERRSTP(900,OUT)
 1200       CONTINUE
          ENDIF
 2000   CONTINUE
        SPTR(NK+1)=NSPTR
        IF(NNTOP.NE.0)THEN
          DO 4000 K=1,NNTOP
            SUCC(NSPTR+K-1)=NTOP(K)
            NTOP(K)=0
 4000     CONTINUE
          NSPTR=NSPTR+NNTOP
          IF(NSPTR.GT.MAXARC)CALL ERRSTP(900,OUT)
        ENDIF
        SPTR(NK+2)=NSPTR
        SPTR(NK+3)=NSPTR
        PPTR(NK+1)=NPPTR
        PPTR(NK+2)=NPPTR
        IF(NSCRTCH.NE.0)THEN
          DO 3000 K=1,NSCRTCH
            PRED(NPPTR+K-1)=SCRATCH(K)
 3000     CONTINUE
          NPPTR=NPPTR+NSCRTCH
          IF(NPPTR.GT.MAXARC)CALL ERRSTP(900,OUT)
        ENDIF
        PPTR(NK+3)=NPPTR
C
C-----NOW DO TOPOLOGICAL SORT AND COMPUTE THE EXTREME
C         CRITICAL
```

```
C-----PATH LENGTHS, USING DMIN AND DMAX.
C
      CALL TSORT(IRTN,SCRATCH)
      IF (IRTN.NE.0)CALL ERRSTP(IRTN,OUT)
      CALL EARLY1(ITIME,NODE,DMIN,SCRATCH)
      LENMIN=SCRATCH(NK+2)
      CALL EARLY1(ITIME,NODE,DMAX,SCRATCH)
      LENMAX=SCRATCH(NK+2)
C
C-----WRITE OUT PROBLEM SUMMARY
C
      WRITE(OUT,10004)NE,NU,LENMIN,LENMAX,TLAB,TSAV
10004 FORMAT("1",////23X,"PROBLEM SUMMARY STATISTICS"/
     *20X,"NUMBER OF ELEMENTS: ",5X,I5/
     *20X,"NUMBER OF UNITS: ",8X,I5/
     *20X,"MINIMUM CPM DURATION: ",3X,I5/
     *20X,"MAXIMUM CPM DURATION: ",3X,I5/
     *20X,"TOTAL LABOR: ",12X,I5/
     *20X,"TOTAL SAVINGS: ",5X,I10/)
      IF(IPRNT.EQ.0)RETURN
C
      ENTRY ECHO1(SCRATCH,OUT)
250   WRITE(OUT,10001)
10001 FORMAT(" ECHO1"/)
      DO 300 I=1,NACT
        SCRATCH(NTOP(I))=I
300   CONTINUE
      GO TO 400
      ENTRY ECHO2(SCRATCH,OUT)
      WRITE(OUT,10002)
10002 FORMAT(" ECHO2"/)
400   DO 500 I=1,NACT
        II=NTOP(I)
        WRITE(OUT,10000)I,II,SCRATCH(II),SPTR(II),PPTR(II),
     *  DMIN(II),
     *                          TL(II),SAV(II),SCRATCH(I)
500   CONTINUE
10000 FORMAT(" ",16I5)
      RETURN
      END
      SUBROUTINE TSORT(IRTN,IFLAG)
C
C-----TO DETERMINE A TOPOLOGICAL ORDERING OF THE ACTIVITIES
C       IN
C-----THE PROBLEM WHOSE DATA IS CONTAINED IN COMMON/PROB/
C
C-----PROBLEM COMMON BLOCK**PROBLEM COMMON BLOCK**PROBLEM
C       COMMON BLOCK**
C
      INTEGER PPTR(250),SPTR(250),TL(250),DMIN(250),
     *  DMAX(250),SAV(250),
     *        NTOP(250),PRED(1000),SUCC(1000)
      COMMON/PROB/PPTR,PRED,SPTR,SUCC,TL,DMIN,DMAX,SAV,NTOP,
     *  NACT,
     *           MAXACT,MAXARC
C
C-----PROBLEM COMMON BLOCK**PROBLEM COMMON BLOCK**PROBLEM
C       COMMON BLOCK**
C
C-----DIMENSION THE WORK SPACE
      DIMENSION IFLAG(1)
C
C-----INITIALIZE COUNT AND FIND THE FIRST ACTIVITY, I.E.
C       THE ONW
C-----WHICH HAS NO PREDECESSORS.
       NCNT=0
      DO 100 K=1,NACT
        IFIRST=PPTR(K)
        ILAST=PPTR(K+1)
        IFLAG(K)=ILAST-IFIRST
        IF(IFLAG(K).EQ.0)THEN
          NTOP(1)=K
          NCNT=NCNT+1
        ENDIF
100   CONTINUE
      IF(NCNT.EQ.1)GO TO 200
      IF(NCNT.EQ.0)THEN
        IRTN=100
      ELSE
        IRTN=200
      ENDIF
      RETURN
200   INODE=1
      INLIST=1
210   IFIRST=SPTR(NTOP(INODE))
      ILAST=SPTR(NTOP(INODE)+1)-1
      IF(IFIRST.LE.ILAST)THEN
        DO 300 KK=IFIRST,ILAST
          IFLAG(SUCC(KK))=IFLAG(SUCC(KK))-1
          IF(IFLAG(SUCC(KK)).EQ.0)THEN
```

```
            INLIST=INLIST+1
            NTOP(INLIST)=SUCC(KK)
          ENDIF
300     CONTINUE
      ENDIF
      INODE=INODE+1
      IF(INODE.LE.INLIST)GO TO 210
      IF(INLIST.EQ.NACT)THEN
        IRTN=0
      ELSE
        IRTN=300
      ENDIF
      RETURN
      END
```

```
C
C
      SUBROUTINE EARLY(TIME,NODE,DUR,ES)
C
C-----TO FIND THE CPM EARLY START SCHEDULE FOR THE PROJECT
C       WHOSE DATA
C-----IS GIVEN IN COMMON/PROB/, USING *TIME* AS THE START
C       TIME
C-----*NODE* AS THE START NODE, AND *DUR* AS THE ACTIVITY
C       DURATIONS
C
C
C-----PROBLEM COMMON BLOCK**PROBLEM COMMON BLOCK**PROBLEM
C       COMMON BLOCK**
C
      INTEGER PPTR(250),SPTR(250),TL(250),DMIN(250),
     *  DMAX(250),SAV(250),
     *        NTOP(250),PRED(1000),SUCC(1000)
      COMMON/PROB/PPTR,PRED,SPTR,SUCC,TL,DMIN,DMAX,SAV,NTOP,
     *  NACT,
     *             MAXACT,MAXARC
C
C-----PROBLEM COMMON BLOCK**PROBLEM COMMON BLOCK**PROBLEM
C       COMMON BLOCK**
C
C
C-----ADDITIONAL DECLARATIONS FOR THIS SUBROUTINE
      INTEGER ES(1),DUR(1),TIME
C
C-----EARLY FINDS THE TOPOLOGICAL ORDER POSITION OF *NODE*
```

```
C-----AND THEN CALCULATES THE ES SCHEDULE FROM THERE.
C-----DETERMINE TOPOLOGICAL ORDER POSITION OF NODE
      DO 50 K=1,NACT
       IF(NTOP(K).EQ.NODE)GO TO 60
50    CONTINUE
      CALL ERRSTP(800,UNIT4)
60    NODE=K
C
      ENTRY EARLY1(TIME,NODE,DUR,ES)
C-----EARLY1 STARTS CALCULATING ES SCHEDULE FROM POSITION
C-----*NODE* OF THE TOPOLOGICAL ORDER RATHER THAN FINDING
C-----*NODE*'S POSITION IN THE ORDER.
C
C-----THUS, A CPM ES SCHEDULE CAN BE OBTAINED FOR A GIVEN
C-----STARTING NODE IN EITHER THE ORIGINAL ACTIVITY
C        NUMVBERING
C-----OR THE TOPOLOGICAL ORDER.
C-----INITIALIZE ES
      DO 100 K=NODE,NACT
        ES(K)=TIME
100   CONTINUE
C-----NOTE THAT ES(I) IS ASSOCIATED WITH ORIGINAL ACTIVITY
C-----I,  NOT WITH THE I-TH ACTIVITY IN TSORT ORDER...
      DO 200 IK=NODE,NACT-1
        K=NTOP(IK)
        IFIN=ES(K)+DUR(K)
        IFIRST=SPTR(K)
        ILAST=SPTR(K+1)-1
        IF(IFIRST.LE.ILAST)THEN
          DO 110 KK=IFIRST,ILAST
            ITEMP=SUCC(KK)
            IF(ES(ITEMP).LT.IFIN)ES(ITEMP)=IFIN
110       CONTINUE
        ENDIF
200   CONTINUE
      RETURN
      END




C
C
      SUBROUTINE PUSH(STACK,MAXREC,NREC,RECSZ,REC)
C
C-----TO PUSH A RECORD OF VARIABLE SIZE ONTO A STACK
C
      INTEGER RECSZ,UNIT1,UNIT2,UNIT3,UNIT4
      COMMON/PARMS/UNIT1,UNIT2,UNIT3,UNIT4
      DIMENSION STACK(MAXREC),REC(RECSZ)
C
C
      NFILL=MAXREC/RECSZ
      IF(NREC.EQ.NFILL)CALL ERRSTP(400,UNIT4)
      LAST=NREC*RECSZ
      DO 100 I=1,RECSZ
        STACK(LAST+I)=REC(I)
100   CONTINUE
      NREC=NREC+1
      RETURN
      END





C
C
      SUBROUTINE POP(STACK,MAXREC,NREC,RECSZ,REC)
C
C-----TO POP A RECORD OF VARIABLE SIZE FROM A STACK
C
      INTEGER RECSZ,UNIT1,UNIT2,UNIT3,UNIT4
      COMMON/PARMS/UNIT1,UNIT2,UNIT3,UNIT4
      DIMENSION STACK(MAXREC),REC(RECSZ)
C
C
      IF(NREC.EQ.0)CALL ERRSTP(500,UNIT4)
      LAST=(NREC-1)*RECSZ
      DO 100 I=1,RECSZ
        REC(I)=STACK(LAST+I)
100   CONTINUE
      NREC=NREC-1
      RETURN
      END





C
C
      SUBROUTINE ERRSTP(KEY,UNIT)
C
C-----TO HANDLE ALL FATAL EXECUTION ERRORS DETECTED BY THE
C       PROGRAM
```

```
C
      INTEGER UNIT
      KEY1=KEY/100
      GO TO (100,200,300,400,500,600,700,800,900,1000)KEY1
      WRITE(UNIT,10000)KEY
10000 FORMAT(" ERRSTP- ",I5," UNRECOGNIZED VALUE FOR KEY")
      GO TO 9999
100   WRITE(UNIT,10100)KEY
10100 FORMAT(" ERRSTP- ",I5," NO ACTIVITY WITH ZERO
     *   PREDECESSORS"/
     *13X," STOP IN TSORT/RUN ABORTED")
      GO TO 9999
200   WRITE(UNIT,10200)KEY
10200 FORMAT(" ERRSTP- ",I5," MORE THAN ONE ACTIVITY WITH
     *   ZERO PRED"/
     *13X," STOP IN TSORT/RUN ABORTED")
      GO TO 9999
300   WRITE(UNIT,10300)KEY
10300 FORMAT(" ERRSTP- ",I5," CYCLE IN PRECEDENCE FILE"/
     *13X," STOP IN TSORT/RUN ABORTED")
      GO TO 9999
400   WRITE(UNIT,10400)KEY
10400 FORMAT(" ERRSTP- ",I5," EXCEEDED STACK CAPACITY"/
     *13X," STOP IN PUSH/RUN ABORTED")
      GO TO 9999
500   WRITE(UNIT,10500)KEY
10500 FORMAT(" ERRSTP- ",I5," ATTEMPT TO POP EMPTY STACK"/
     *13X,"STOP IN POP/RUN ABORTED")
      GO TO 9999
600   WRITE(UNIT,10600)KEY
10600 FORMAT(" ERRSTP- ",I5," RNET ERROR RETURN"/
     *13X,"STOP IN SOLVE/RUN ABORTED")
      GO TO 9999
700   WRITE(UNIT,10700)KEY
10700 FORMAT(" ERRSTP- ",I5," RNET VALUE .NE. PRIMAL VALUE"/
     *13X,"STOP IN SOLVE/RUN ABORTED")
      GO TO 9999
800   WRITE(UNIT,10800)KEY
10800 FORMAT(" ERRSTP- ",I5," NODE NOT FOUND IN NTOP BY ",
     *"SUBROUTINE EARLY"/13X,"STOP IN EARLY/RUN ABORTED")
      GO TO 9999
900   WRITE(UNIT,10900)KEY
10900 FORMAT(" ERRSTP- ",I5," PROBLEM SIZE EXCEEDS",
     *" PROGRAM LIMITS"/13X," STOP IN DATIN/RUN ABORTED")
      GO TO 9999
1000  WRITE(UNIT,11000)KEY
11000 FORMAT(" ERRSTP- ",I5," MISSING ACTIVITY DATA IN",
     *" SPECS FILE"/" STOP IN DATIN/RUN ABORTED")
      GO TO 9999
9999  STOP
```

```
      END




C
C
      FUNCTION PROJECT(LX,RX,LSLOP,RSLOP,LY,RY)
C
C-----FUNCTION SUBROUTINE TO COMPUTED THE PROJECTED VALUE
C       OF LAMBDA
C
      REAL LX,LSLOP,LY
      PROJECT=(RY-LY-LX*LSLOP+RX*RSLOP)/(RSLOP-LSLOP)
      RETURN
      END
```

APPENDIX B

PROGRAM SCHEDULE LISTING

```
      PROGRAM SKEDULE(TAPE1,TAPE2,INPUT,OUTPUT,
     *                TAPE5=INPUT,TAPE6=OUTPUT)
C
C-----THIS PROGRAM DETERMINES A RESOURCE FEASIBLE SCHEDULE
C       FOR THE
C_____OUTFIT PLANNING PROBLEM.  THE INPUT DATA FILE IS
C       PROVIDED
C-----BY PROGRAM *EXTRACT*.  THE RESOURCE PROFILE IS
C       SPECIFIED
C-----IN THE *INPUT* FILE.
C
C-----------------------------------------------------------
C       ------
C
C     INPUT:
C       (1) PROBLEM FILE--"TAPE1"
C       (2) RESOURCE PROFILE--"INPUT"
C
C     OUTPUT
C       (1) PROBLEM SUMMARY--"OUTPUT"
C       (2) SOLUTION SUMMARY--"OUTPUT"
C       (3) DETAILED SOLUTION--"TAPE2"
C
C     USES ROUTINES
C       (1) PROBIN
C       (2) SELECT
C       (3) CSETUP
C       (4) CSETAUG
C
C-----------------------------------------------------------
C       ------
C
C----------------------PROBLEM PARAMETERS-------------------
C       --
C
      INTEGER SPTR(250),SUCC(1000),PPTR(250),PRED(1000),
     *  DMIN(250),
     *        DMAX(250),TL(250),SAV(250),PROFIL(25,3)
      COMMON/PROB/NACT,TLAB,TVAL,NINC,TRES,SPTR,SUCC,PPTR,
     *  PRED,
     *            DMIN,DMAX,TL,SAV,PROFIL
C
C----------------------PROBLEM PARAMETERS-------------------
C       --
C----------------------KNAPSACK PARAMETERS------------------
C       --
C
      INTEGER XS(250),PTR(250)
      DIMENSION RREQ(250),VAL(250)
      COMMON/KNAP/BRHS,BOU,S,RL,RLB,BUDGE,RREQ,VAL,XS,PTR
```

```
C
C------------------------KNAPSACK PARAMETERS------------------
C       --
C------------------------SOLUTION PARAMETERS------------------
C       --
C
      INTEGER IFLAG(250),ASET(250),CSET(250),SSET(250),
     *  ES(250),
     *        LS(250),DUR(250),MD(250)
      INTEGER PINC,TIME,TNEXT,TDROP,TMAX
      COMMON/SOLNC/PINC,NASET,NCSET,NSSET,TIME,TNEXT,TDROP,
     *  TMAX,
     *             RASET,RAVAIL,RADROP,RUDROP,NSKED,SVAL,
     *  SLAB,
     *             NDEC,NKNAP,TKNAP,TTOT,SFAC1,SFAC2
      COMMON/SOLNA/IFLAG,ASET,CSET,SSET,ES,LS,DUR,MD
C
C------------------------SOLUTION PARAMETERS------------------
C       --
C
C--------------------------OTHER PARAMETERS-------------------
C       --
C
      INTEGER UNIT1,UNIT2,UNIT3,UNIT4
      COMMON/PARMS/UNIT1,UNIT2,UNIT3,UNIT4
      DATA S,RL,BOU/0.,10E10,-99/,UNIT1,UNIT2,UNIT3,UNIT4/1,
     *  2,5,6/
      DATA SCAF1,SCAF2/10.E+4,10./
C-------------------------------------------------------------
C       ------
C-------------------------------------------------------------
C       ------
C
C-----BEGIN SCHEDULING
C
C-----GET PARAMETERS AND DATA
      WRITE(UNIT4,10001)
10001 FORMAT(" ENTER IPRNT")
      READ(UNIT3,*)IPRNT
      CALL PROBIN(IPRNT)
      WRITE(UNIT4,10002)
10002 FORMAT(" ENTER RESOURCE PROFILE AS TRIPLES, T1,T2,
     *  LEVEL")
      I=1
      NDOWN=0
      TRES=0
100   READ(UNIT3,*,END=200)IT1,IT2,LEV
      PROFIL(I,1)=IT1
      PROFIL(I,2)=IT2
      PROFIL(I,3)=LEV
      TRES=TRES+LEV*(IT2-IT1)
      IF(I.EQ.1)GO TO 110
      IF(LEV.LT.PROFIL(I-1,3))NDOWN=NDOWN+1
110   I=I+1
      GO TO 100
200   CONTINUE
      NINC=I-1
      IF(NDOWN.GT.2)CALL ERRSTP(100,UNIT4)
      TDROP=PROFIL(NINC-1,1)
      RADROP=PROFIL(NINC-1,3)
      TMAX=PROFIL(NINC,1)
      IF(IPRNT.NE.0)WRITE(*,90000) TDROP,RADROP,TMAX
90000 FORMAT(/" TDROP,RADROP,TMAX",I5,F10.2,I5)
C
C-----INITIALIZE THE RESOURCE CONSTRAINED HEURISTIC
C
      TSTART=SECOND()
      PINC=1
      NSKED=0
      NDEC=0
      SVAL=0
      SLAB=0
      NCSET=1
      CSET(1)=1
      NASET=0
      TIME=0
C-----INITIALIZE NUMBER OF PREDECESSORS
      DO 300 I=1,NACT
        IFLAG(I)=PPTR(I)-PPTR(I+1)
300   CONTINUE
      IF(IPRNT.GT.2)WRITE(*,90002)(I,IFLAG(I),I=1,NACT)
90002 FORMAT(/" INITIAL IFLAG"/(2I10))
C
C-----BEGIN MAIN LOOP
C-----IF CSET IS EMPTY WE ARE DONE
      IF(NCSET.EQ.0)GO TO 9000
1000  NDEC=NDEC+1
      IF(IPRNT.NE.0)WRITE(*,90001)TIME,NCSET,(CSET(I),I=1,
     *  NCSET)
90001 FORMAT(/" DECISION TIME= ",I5," NCSET= ",I5/(15I5))
      CALL SELECT(IPRNT)
      IF(NSSET.NE.0)CALL CSETUP(IPRNT)
2000  ITM1=TMAX
      IF(NASET.NE.0)ITM1=IFLAG(ASET(NASET))
      ITM2=TMAX
      IF(PINC.LT.NINC)THEN
        IF (PROFIL(PINC+1,3).GT.PROFIL(PINC,3))ITM2=
     *  PROFIL(PINC,2)
      ENDIF
      TIME=MIN(ITM1,ITM2)
      CALL CSETAUG(IPRNT)
      IF(NCSET.NE.0)GO TO 1000
      IF(NASET.NE.0)GO TO 2000
9000  CONTINUE
C
C-----HEURISTIC IS DONE
C
      TTOT=SECOND()-TSTART
      NDEF=NACT-NSKED
C-----WRITE SOLUTION SUMMARY
      R1=(SLAB)/TRES
      R2=(SVAL)/TVAL
      R3=FLOAT(NDEF)/NACT
      WRITE(UNIT4,10003)TMAX,TRES,SLAB,R1,TVAL,SVAL,R2,
     *                  NDEF,R3,NDEC,NKNAP,TKNAP,TTOT
10003 FORMAT("1",///32X,"SOLUTION SUMMARY"/32X,16("-")/
     *21X,"PROFILE DURATION",16("."),I5/
     *21X,"MANDAYS AVAILABLE",8("."),F12.2/
     *21X,"MANDAYS USED",13("."),F12.2/
     *21X,"LABOR UTILAZATION FACTOR......",F6.4/
     *21X,"VALUE SELECTED",11("."),F12.0/
     *21X,"VALUE SCHEDULED",10("."),F12.0/
     *21X,"SCHEDULE PERFORMANCE FACTOR....",F6.4/
     *21X,"NUMBER OF DEFERRED ACTIVITIES...",I5/
     *21X,"FRACTION OF DEFERRED ACTIVITIES.",F5.4/
     *21X,37("-")/
     *21X,"NUMBER OF SCHEDULING DECISIONS..",I5/
     *21X,"NUMBER OF KNAPSACK PROBLEMS.....",I5/
     *21X,"KNAPSACK TIME",16("."),F8.2/
     *21X,"TOTAL TIME",19("."),F8.2////)
      WRITE(UNIT4,10004)
10004 FORMAT("1"///" **** RUN TERMINATED NORMALLY ****")
      STOP
      END




C
C
C
      SUBROUTINE CSETUP(IPRNT)
C
C-----THIS SUBROUTINE SHIFTS SELECTED ACTIVITIES FROM THE
C-----JUST SCHEDULED SET TO THE ACTIVE SET
C
C
C------------------------------------------------------------
C         ---------
```

```
C-----------------------PROBLEM PARAMETERS------------------
C       --
C
      INTEGER SPTR(250),SUCC(1000),PPTR(250),PRED(1000),
     *  DMIN(250),
     *        DMAX(250),TL(250),SAV(250),PROFIL(25,3)
      COMMON/PROB/NACT,TLAB,TVAL,NINC,TRES,SPTR,SUCC,PPTR,
     *  PRED,
     *            DMIN,DMAX,TL,SAV,PROFIL
C
C-----------------------PROBLEM PARAMETERS------------------
C       --
C-----------------------KNAPSACK PARAMETERS-----------------
C       --
C
      INTEGER XS(250),PTR(250)
      DIMENSION RREQ(250),VAL(250)
      COMMON/KNAP/BRHS,BOU,S,RL,RLB,BUDGE,RREQ,VAL,XS,PTR
C
C-----------------------KNAPSACK PARAMETERS-----------------
C       --
C-----------------------SOLUTION PARAMETERS-----------------
C       --
C
      INTEGER IFLAG(250),ASET(250),CSET(250),SSET(250),
     *  ES(250),
     *        LS(250),DUR(250),MD(250)
      INTEGER PINC,TIME,TNEXT,TDROP,TMAX
      COMMON/SOLNC/PINC,NASET,NCSET,NSSET,TIME,TNEXT,TDROP,
     *  TMAX,
     *             RASET,RAVAIL,RADROP,RUDROP,NSKED,SVAL,
     *  SLAB,
     *             NDEC,NKNAP,TKNAP,TTOT,SFAC1,SFAC2
      COMMON/SOLNA/IFLAG,ASET,CSET,SSET,ES,LS,DUR,MD
C
C-----------------------SOLUTION PARAMETERS-----------------
C       --
C----------------------------------------------------------
C       ---------
      IF(NSSET.EQ.0)RETURN
      IF(IPRNT.GT.1)THEN
        WRITE(*,90001)(ASET(I),I=1,NASET)
        WRITE(*,90002)(CSET(I),I=1,NCSET)
      ENDIF
90001 FORMAT(/" CSETUP: ASET"/(15I5))
90002 FORMAT(/" CSETUP: CSET"/(15I5))
C
C-----REMOVE SELECTED ACTIVITIES FROM CSET AND ADD THEM TO
C       ASET,
C-----KEEPING ASET IN DECREASING ORDER OF COMPLETION TIME
```

```
      K=1
1010  IF(K.GT.NCSET)GO TO 1100
      IF(CSET(K).LT.0)THEN
        NTEMP=-CSET(K)
        NCSET=NCSET-1
        SVAL=SVAL+SAV(NTEMP)
        SLAB=SLAB+TL(NTEMP)
        NSKED=NSKED+1
        DO 1020 KK=K,NCSET
          CSET(KK)=CSET(KK+1)
1020    CONTINUE
        DO 1030 KK=1,NASET
          IF(IFLAG(NTEMP).GT.IFLAG(ASET(KK)))THEN
            IHOLD=ASET(KK)
            ASET(KK)=NTEMP
            NTEMP=IHOLD
          ENDIF
1030    CONTINUE
        NASET=NASET+1
        ASET(NASET)=NTEMP
      ELSE
        K=K+1
      ENDIF
      GO TO 1010
1100  CONTINUE
      IF(IPRNT.GT.1)THEN
        WRITE(*,90001)(ASET(I),I=1,NASET)
        WRITE(*,90002)(CSET(I),I=1,NCSET)
      ENDIF
C-----CSET AND ASET ARE NOW UPDATED FOR THE JUST SCHEDULED
C       ACTIVITIES
      RETURN
      END




C
C
C
      SUBROUTINE CSETAUG(IPRNT)
C
C-----THIS SUBROUTINE AUGMENTS CSET BY LOOKING AT THE
C       SUCCESSORS OF
C-----THOSE ACTIVITIES WHICH COMPLETE AT *TIME* AND IF THE
C       SUCCESSORS
C-----HAVE NO MORE UNSCHEDULED PREDECESSORS, THEY ARE ADDED
C       TO CSET.
```

```
C
C
C-----------------------------------------------------------------------
C        ---------
C-----------------------PROBLEM PARAMETERS------------------
C        --
C
      INTEGER SPTR(250),SUCC(1000),PPTR(250),PRED(1000),
     *  DMIN(250),
     *        DMAX(250),TL(250),SAV(250),PROFIL(25,3)
      COMMON/PROB/NACT,TLAB,TVAL,NINC,TRES,SPTR,SUCC,PPTR,
     *  PRED,
     *            DMIN,DMAX,TL,SAV,PROFIL
C
C-----------------------PROBLEM PARAMETERS------------------
C        --
C-----------------------KNAPSACK PARAMETERS-----------------
C        --
C
      INTEGER XS(250),PTR(250)
      DIMENSION RREQ(250),VAL(250)
      COMMON/KNAP/BRHS,BOU,S,RL,RLB,BUDGE,RREQ,VAL,XS,PTR
C
C-----------------------KNAPSACK PARAMETERS-----------------
C        --
C-----------------------SOLUTION PARAMETERS-----------------
C        --
C
      INTEGER IFLAG(250),ASET(250),CSET(250),SSET(250),
     *  ES(250),
     *        LS(250),DUR(250),MD(250)
      INTEGER PINC,TIME,TNEXT,TDROP,TMAX
      COMMON/SOLNC/PINC,NASET,NCSET,NSSET,TIME,TNEXT,TDROP,
     *  TMAX,
     *             RASET,RAVAIL,RADROP,RUDROP,NSKED,SVAL,
     *  SLAB,
     *             NDEC,NKNAP,TKNAP,TTOT,SFAC1,SFAC2
      COMMON/SOLNA/IFLAG,ASET,CSET,SSET,ES,LS,DUR,MD
C
C-----------------------SOLUTION PARAMETERS-----------------
C        --
C-----------------------------------------------------------------------
C        ---------
      IF(IPRNT.GT.1)WRITE(*,90001)TIME,NASET
90001 FORMAT(/" BEGIN CSETAUG, TIME= ",I5," NASET=",I5)
      IF(NASET.EQ.0)RETURN
1200  IF(IFLAG(ASET(NASET)).GT.TIME)RETURN
      IF(IPRNT.GT.1)THEN
        WRITE(*,90002)ASET(NASET),IFLAG(ASET(NASET))
      ENDIF
```

```
90002 FORMAT(/" ACTIVE NODE",I5," FINISHES AT TIME",I5)
      IFIRST=SPTR(ASET(NASET))
      ILAST=SPTR(ASET(NASET)+1)-1
      IF(IFIRST.LE.ILAST)THEN
        DO 1210 KK=IFIRST,ILAST
          KKS=SUCC(KK)
          IFLAG(KKS)=IFLAG(KKS)+1
          IF(IFLAG(KKS).EQ.0)THEN
            NCSET=NCSET+1
            CSET(NCSET)=KKS
      IF(IPRNT.GT.2)WRITE(*,90003)KKS
90003 FORMAT(/" ADD NODE",I5," TO CANDIDATE SET")
          ENDIF
1210    CONTINUE
      ENDIF
      NASET=NASET-1
      IF(NASET.NE.0)GO TO 1200
      RETURN
      END




C
C
C
      SUBROUTINE SELECT(IPRNT)
C
C-----THIS SUBROUTINE MANAGES THE SCHEDULING DECISION RULE
C       THAT IS
C-----APPLIED WHENEVER THERE ARE RESOURCES TO BE ALLOCATED.
C
C
C
C-----------------------------------------------------------
C       ---------
C----------------------PROBLEM PARAMETERS-------------------
C       --
C
      INTEGER SPTR(250),SUCC(1000),PPTR(250),PRED(1000),
     *  DMIN(250),
     *        DMAX(250),TL(250),SAV(250),PROFIL(25,3)
      COMMON/PROB/NACT,TLAB,TVAL,NINC,TRES,SPTR,SUCC,PPTR,
     *  PRED,
     *            DMIN,DMAX,TL,SAV,PROFIL
C
C----------------------PROBLEM PARAMETERS-------------------
C       --
```

```
C-----------------------KNAPSACK PARAMETERS-----------------
C       --
C
      INTEGER XS(250),PTR(250)
      DIMENSION RREQ(250),VAL(250)
      COMMON/KNAP/BRHS,BOU,S,RL,RLB,BUDGE,RREQ,VAL,XS,PTR
C
C-----------------------KNAPSACK PARAMETERS-----------------
C       --
C-----------------------SOLUTION PARAMETERS-----------------
C       --
C
      INTEGER IFLAG(250),ASET(250),CSET(250),SSET(250),
     *  ES(250),
     *        LS(250),DUR(250),MD(250)
      INTEGER PINC,TIME,TNEXT,TDROP,TMAX
      COMMON/SOLNC/PINC,NASET,NCSET,NSSET,TIME,TNEXT,TDROP,
     *  TMAX,
     *            RASET,RAVAIL,RADROP,RUDROP,NSKED,SVAL,
     *  SLAB,
     *            NDEC,NKNAP,TKNAP,TTOT,SFAC1,SFAC2
      COMMON/SOLNA/IFLAG,ASET,CSET,SSET,ES,LS,DUR,MD
C
C-----------------------SOLUTION PARAMETERS-----------------
C       --
      IF(IPRNT.GT.2)WRITE(*,90000)
90000 FORMAT(//" START SUBROUTINE SELECT")
      NSSET=0
      IBRNCH=1
      IF(NCSET.EQ.0)RETURN
      CALL SORT1(CSET,NCSET)
      ICSET=CSET(1)
C-----DROP ACTIVITIES WHICH CANNOT BE DONE BY TMAX
      IF(IPRNT.GT.2)WRITE(*,90004)
90004 FORMAT(/" CALL EARLY")
      CALL EARLY(ICSET,TIME,DMIN,ES,IFLAG)
      IF(IPRNT.GT.2)WRITE(*,90005)(I,ES(I),I=ICSET,NACT)
90005 FORMAT(/" EARLY RETURN:"/(3X,2I8))
      DO 30 I=ICSET,NACT
        IF(ES(I)+DMIN(I).LE.TMAX)GO TO 30
        IF(IPRNT.GT.2)WRITE(*,90002)I,ES(I),DMIN(I)
90002   FORMAT(/" DROP ACTIVITY ",I5," STARTS AT",I5,"
     *  DURATION",I5)
        IFLAG(I)=IFLAG(I)-1
        DUR(I)=0
        DMIN(I)=0
        DMAX(I)=0
30    CONTINUE
      DO 45 I=1,NCSET
        IF(IFLAG(CSET(I)).EQ.0) GO TO 45
```

```
          NCSET=NCSET-1
          IF(I.LE.NCSET)THEN
            DO 40 K=I,NCSET
              CSET(K)=CSET(K+1)
40          CONTINUE
          ENDIF
45      CONTINUE
C
C-----NOW SCHEDULE ALL ACTIVITIES WITH ZERO RESOURCE REQMT
C--------------------------------------------------------------
C         ---------
C----- SELECT ALL ACTIVITIES WHICH REQUIRE NO RESOURCES
50      IFLG=0
        DO 60 I=1,NCSET
          IF(TL(CSET(I)).EQ.0)THEN
            IFLG=IFLG+1
            DUR(CSET(I))=DMIN(CSET(I))
            IFLAG(CSET(I))=TIME+DMIN(CSET(I))
            CSET(I)=-CSET(I)
          ENDIF
60      CONTINUE
        IF(IFLG.EQ.0)GO TO 70
        NSSET=IFLG
        CALL CSETUP(IPRNT)
C-----IF ANY ACTIVITY HAS A ZERO DURATION, THEN WE HAVE TO
C         AUGMENT
C-----CSET OR ELSE WE ARE DONE.
        CALL CSETAUG(IPRNT)
        GO TO 50
70      CONTINUE
C
C-----NOW GET THE RESOURCE PARAMETERS
C
        RASET=0
        IF(NASET.EQ.0)GO TO 100
        DO 80 I=1,NASET
          RASET=RASET+FLOAT(TL(ASET(I)))/DUR(ASET(I))
80      CONTINUE
100     T1=PROFIL(PINC,1)
        T2=PROFIL(PINC,2)
        RES=PROFIL(PINC,3)
        IF(T1.LE.TIME.AND.TIME.LT.T2)GO TO 200
        PINC=PINC+1
        IF(PINC.GT.NINC)CALL ERRSTP(200,UNIT4)
        GO TO 100
200     RAVAIL=RES-RASET
        IF(IPRNT.GT.1)WRITE(*,90003)RES,RASET,RAVAIL
90003 FORMAT(/" PROFILE=", F10.2," ACTIVE= ",F10.2," AVAIL=
     *   ",F10.2)
C-----NOW SET TENTATIVE DURATIONS
```

```
      CALL SORT1(CSET,NCSET)
      ICSET=CSET(1)
250   CONTINUE
      IF(IBRNCH.EQ.0)THEN
      DO 300 I=ICSET,NACT
        IF(IFLAG(I).LE.0)THEN
          DUR(I)=DMIN(I)+((TMAX-TIME)/TMAX)*(DMAX(I)-DMIN(I)
     *  )
        ENDIF
300   CONTINUE
      IF(IPRNT.GT.2)WRITE(*,90008)(I,DUR(I),I=ICSET,NACT)
90008 FORMAT(/" TENTATIVE DURATIONS"/(2I10))
      CALL EARLY(ICSET,TIME,DUR,ES,IFLAG)
      CALL LATE(ICSET,TMAX,DUR,LS,IFLAG)
      IF(IPRNT.GT.2)WRITE(*,90009)(I,ES(I),LS(I),I=ICSET,
     *  NACT)
90009 FORMAT(/" TENTATIVE ES AND LS"/(3I10))
      DO 400 I=ICSET,NACT
        IF(ES(I).GT.LS(I))THEN
          JFLG=JFLG+1
          IF(DMAX(I).EQ.DMIN(I))GO TO 400
          IFLG=IFLG+1
          DMAX(I)=DMIN(I)+0.5*(DMAX(I)-DMIN(I))
        ENDIF
400   CONTINUE
      IF(IFLG.NE.0)GO TO 250
      IF(JFLG.NE.0)CALL ERRSTP(300,UNIT4)
      ELSE
C-----ALTERNATIVE TENTATIVE DURATIONS
      DO 430 I=ICSET,NACT
430   DUR(I)=DMIN(I)
      DO 440 I=1,NCSET
      DUR(CSET(I))=DMAX(CSET(I))
440   IF(TIME+DUR(CSET(I)).GT.TMAX)DUR(CSET(I))=TMAX-TIME
      CALL LATE(ICSET,TMAX,DUR,LS,IFLAG)
      ENDIF
C-----NOW READY TO MAKE SELECTION
      IF(IPRNT.NE.0)WRITE(*,90010)
90010 FORMAT(/" READY TO MAKE NONTRIVIAL SELECTION")
      NSSET=0
C-----CAN ALL OR NONE BE SELECTED?
      RCSET=0
      IFLG=0
C-----COMPUTE CANDIDATE SET RESOURCE REQUIREMENT
      DO 500 I=1,NCSET
        RREQ(I)=FLOAT(TL(CSET(I)))/DUR(CSET(I))
        RCSET=RCSET+RREQ(I)
        IF(RREQ(I).LE.RAVAIL)IFLG=1
500   CONTINUE
      IF(IPRNT.GT.1)THEN
```

```
        WRITE(*,90011)RCSET,(CSET(I),RREQ(I),I=1,NCSET)
      ENDIF
90011 FORMAT(/" CANDIDATE REQUIREMENTS",F10.2/(I5,F10.2))
      IF(IFLG.EQ.0)RETURN
      IF(RCSET.GT.RAVAIL)GO TO 1000
      DO 600 I=1,NCSET
        PTR(I)=I
        XS(I)=1
600   CONTINUE
      RSSET=RCSET
      GO TO 2000
C-----SET UP AND SOLVE THE KNAPSACK PROBLEM
1000  CONTINUE
      IF(IPRNT.GT.1)WRITE(*,90012)
90012 FORMAT(/" CALL PREKNAP")
      CALL PREKNAP
      TS=SECOND()
      NKNAP=NKNAP+1
      IF(IPRNT.GT.1)WRITE(*,90013)(I,VAL(I),I=1,NCSET)
90013 FORMAT(/" CALL KNAP WITH COEF:"/(I5,F10.2))
      CALL KNAP(VAL,RREQ,RAVAIL,NCSET,BOU,S,RL,XS,RLB,BUDGE,
     *  U)
      IF(IPRNT.GT.1)WRITE(*,90014)(PTR(I),XS(I),I=1,NCSET)
90014 FORMAT(/" KNAPSACK SOLUTION:"/(2I5))
      TKNAP=TKNAP+SECOND()-TS
C-----DECODE XS TO GET SSET, CHECK FOR RESOURCE
C-----VIOLATIONS IN FUTURE PERIODS.
      RSSET=BUDGE
2000  RUDROP=0.
      DO 2100 I=1,NCSET
        IF(XS(I).EQ.1)THEN
          ITEMP=CSET(PTR(I))
          IF(TIME+DUR(ITEMP).GT.TDROP)RUDROP=RUDROP+RREQ(I)
        ENDIF
2100  CONTINUE
      IF(RUDROP.LE.RADROP)GO TO 3000
      CALL ADJUST(IPRNT)
      ITER=1
2200  TS=SECOND()
      NKNAP=NKNAP+1
      CALL KNAP(VAL,RREQ,RAVAIL,NCSET,BOU,S,RL,XS,RLB,BUDGE,
     *  U)
      TKNAP=TKNAP+SECOND()-TS
      RUDROP=0.
      DO 2300 I=1,NCSET
        IF(XS(I).EQ.1)THEN
          ITEMP=CSET(PTR(I))
          IF(TIME+DUR(ITEMP).GT.TDROP)RUDROP=RUDROP+RREQ(I)
        ENDIF
2300  CONTINUE
```

```
      IF(RUDROP.LE.RADROP)GO TO 3000
      CALL PENALTY(ITER)
      ITER=ITER+1
      GO TO 2200
3000  NSSET=0
      DO 3100 I=1,NCSET
        IF(XS(I).EQ.1)THEN
          CSET(PTR(I))=-CSET(PTR(I))
          NSSET=NSSET+1
          SSET(NSSET)=-CSET(PTR(I))
        ENDIF
3100  CONTINUE
      CALL SETDUR(IPRNT)
      IF(IPRNT.NE.0)WRITE(*,90015)(SSET(I),I=1,NSSET)
90015 FORMAT(/" SCHEDULED SET:"/(15I5))
      IF(IPRNT.GT.1)WRITE(*,90016)(DUR(SSET(I)),I=1,NSSET)
90016 FORMAT(/" SSET DURATIONS:"/(15I5))
      DO 3200 I=1,NSSET
        IFLAG(SSET(I))=TIME+DUR(SSET(I))
3200  CONTINUE
      IF(IPRNT.NE.0)WRITE(*,90001)
90001 FORMAT(//" END SUBROUTINE SELECT")
      RETURN
      END





C
C
C
      SUBROUTINE PREKNAP
C
C-----THIS SUBROUTINE COMPUTES THE INITIAL COEFFICIENTS FOR
C-----THE KNAPSACK PROBLEM.
C
C-------------------------------------------------------------------
C         --------
C------------------------PROBLEM PARAMETERS-------------------------
C         --
C
      INTEGER SPTR(250),SUCC(1000),PPTR(250),PRED(1000),
     *  DMIN(250),
     *        DMAX(250),TL(250),SAV(250),PROFIL(25,3)
      COMMON/PROB/NACT,TLAB,TVAL,NINC,TRES,SPTR,SUCC,PPTR,
     *  PRED,
     *            DMIN,DMAX,TL,SAV,PROFIL
C
```

```
C------------------------PROBLEM PARAMETERS------------------
C       --
C------------------------KNAPSACK PARAMETERS-----------------
C       --
C
      INTEGER XS(250),PTR(250)
      DIMENSION RREQ(250),VAL(250)
      COMMON/KNAP/BRHS,BOU,S,RL,RLB,BUDGE,RREQ,VAL,XS,PTR
C
C------------------------KNAPSACK PARAMETERS-----------------
C       --
C------------------------SOLUTION PARAMETERS-----------------
C       --
C
      INTEGER IFLAG(250),ASET(250),CSET(250),SSET(250),
     *  ES(250),
     *          LS(250),DUR(250),MD(250)
      INTEGER PINC,TIME,TNEXT,TDROP,TMAX
      COMMON/SOLNC/PINC,NASET,NCSET,NSSET,TIME,TNEXT,TDROP,
     *  TMAX,
     *               RASET,RAVAIL,RADROP,RUDROP,NSKED,SVAL,
     *  SLAB,
     *               NDEC,NKNAP,TKNAP,TTOT,SFAC1,SFAC2
      COMMON/SOLNA/IFLAG,ASET,CSET,SSET,ES,LS,DUR,MD
C
C------------------------SOLUTION PARAMETERS-----------------
C       --
C-----------------------------------------------------------
C       --------
C
C-----DETERMINE TNEXT
      TNEXT=IFLAG(ASET(NASET))
      DO 100 I=1,NCSET
        TNEXT=MIN(TNEXT,TIME+DUR(CSET(I)))
100   CONTINUE
      DO 200 I=1,NCSET
        MD(I)=LS(CSET(I))-TNEXT
        IF(MD(I).LT.0)THEN
          DUR(CSET(I))=MIN(DMAX(CSET(I)),DMIN(CSET(I))-MD(I)
     *  )
          MD(I)=0
        ENDIF
        VAL(I)=MD(I)*SFAC1 + SAV(CSET(I))
        PTR(I)=I
200   CONTINUE
      CALL  SORT6(VAL,RREQ,PTR,NCSET)
      RETURN
      END
```

```
C
C
C
      SUBROUTINE ADJUST(IPRNT)
C
C-----THIS SUBROUTINE ATTEMPTS TO ADJUST ACTIVITY DURATIONS
C        TO
C-----PERMIT A FEASIBLE SCHEDULE SET TO BE CHOSEN USING
C-----WITHOUT HAVING TO RESORT TO THE PENALTY METHOD.
C
C-------------------------------------------------------------
C        ----------
C
C----------------------PROBLEM PARAMETERS---------------------
C        --
C
      INTEGER SPTR(250),SUCC(1000),PPTR(250),PRED(1000),
     *   DMIN(250),
     *          DMAX(250),TL(250),SAV(250),PROFIL(25,3)
      COMMON/PROB/NACT,TLAB,TVAL,NINC,TRES,SPTR,SUCC,PPTR,
     *   PRED,
     *                DMIN,DMAX,TL,SAV,PROFIL
C
C----------------------PROBLEM PARAMETERS---------------------
C        --
C----------------------KNAPSACK PARAMETERS--------------------
C        --
C
      INTEGER XS(250),PTR(250)
      DIMENSION RREQ(250),VAL(250)
      COMMON/KNAP/BRHS,BOU,S,RL,RLB,BUDGE,RREQ,VAL,XS,PTR
C
C----------------------KNAPSACK PARAMETERS--------------------
C        --
C----------------------SOLUTION PARAMETERS--------------------
C        --
C
      INTEGER IFLAG(250),ASET(250),CSET(250),SSET(250),
     *   ES(250),
     *          LS(250),DUR(250),MD(250)
      INTEGER PINC,TIME,TNEXT,TDROP,TMAX
      COMMON/SOLNC/PINC,NASET,NCSET,NSSET,TIME,TNEXT,TDROP,
     *   TMAX,
     *                RASET,RAVAIL,RADROP,RUDROP,NSKED,SVAL,
     *   SLAB,
     *                NDEC,NKNAP,TKNAP,TTOT,SFAC1,SFAC2
      COMMON/SOLNA/IFLAG,ASET,CSET,SSET,ES,LS,DUR,MD
C
C----------------------SOLUTION PARAMETERS--------------------
C        --
```

```
C
C
C-------------------------------------------------------------
C        ----------
C-------------------------------------------------------------
C        ----------
C
      DO 100 I=1,NCSET
        J=CSET(I)
        IF(TIME+DUR(J).GT.TDROP)THEN
          IF(TIME+DMIN(J).LE.TDROP)THEN
            DUR(J)=TDROP-TIME
            TNEXT=MIN(TNEXT,TDROP)
          ELSE
            DUR(J)=MIN(DMAX(J),DUR(J)-ES(J))
          ENDIF
        ENDIF
100   CONTINUE
      DO 200 I=1,NCSET
        MD(I)=LS(CSET(I))-TNEXT
        IF(MD(I).LT.0)MD(I)=0
        VAL(I)=MD(I)*SFAC1 + SAV(CSET(I))
        RREQ(I)=FLOAT(TL(CSET(I)))/DUR(CSET(I))
        PTR(I)=I
200   CONTINUE
      CALL  SORT5(VAL,RREQ,PTR,NCSET)
      RETURN
      END




C
C
C
      SUBROUTINE  PENALTY
C
C-----THIS SUBROUTINE CALCULATES AND APPLIES A GRADUAL
C       PENALTY
C-----TO THE ACTIVITIES WHOSE FINISH TIME EXCEED TDROP.
C       THIS
C-----IS REPEATED UNTIL A SELECTION IS MADE WHICH IS
C       FEASIBLE
C-----AT TIME TDROP.
C
C-------------------------------------------------------------
C        ---------
C
```

```
C
C-----------------------PROBLEM PARAMETERS------------------
C      --
C
      INTEGER SPTR(250),SUCC(1000),PPTR(250),PRED(1000),
     *  DMIN(250),
     *        DMAX(250),TL(250),SAV(250),PROFIL(25,3)
      COMMON/PROB/NACT,TLAB,TVAL,NINC,TRES,SPTR,SUCC,PPTR,
     *  PRED,
     *            DMIN,DMAX,TL,SAV,PROFIL
C
C-----------------------PROBLEM PARAMETERS------------------
C      --
C-----------------------KNAPSACK PARAMETERS-----------------
C      --
C
      INTEGER XS(250),PTR(250)
      DIMENSION RREQ(250),VAL(250)
      COMMON/KNAP/BRHS,BOU,S,RL,RLB,BUDGE,RREQ,VAL,XS,PTR
C
C-----------------------KNAPSACK PARAMETERS-----------------
C      --
C-----------------------SOLUTION PARAMETERS-----------------
C      --
C
      INTEGER IFLAG(250),ASET(250),CSET(250),SSET(250),
     *  ES(250),
     *        LS(250),DUR(250),MD(250)
      INTEGER PINC,TIME,TNEXT,TDROP,TMAX
      COMMON/SOLNC/PINC,NASET,NCSET,NSSET,TIME,TNEXT,TDROP,
     *  TMAX,
     *             RASET,RAVAIL,RADROP,RUDROP,NSKED,SVAL,
     *  SLAB,
     *             NDEC,NKNAP,TKNAP,TTOT,SFAC1,SFAC2
      COMMON/SOLNA/IFLAG,ASET,CSET,SSET,ES,LS,DUR,MD
C
C-----------------------SOLUTION PARAMETERS-----------------
C      --
C
C----------------------------------------------------------
C      ----------
C----------------------------------------------------------
C      ---------
C
      LAMBDA=SFAC2*2**ITER
      DO 100 I=1,NCSET
        DTEST=TDROP-TIME
        II=CSET(I)
        IF(DUR(II).GT.DTEST)THEN
          VAL(I)=MD(I)*SFAC1 + SAV(CSET(I))-LAMBDA
          RREQ(I)=FLOAT(TL(CSET(I)))/DUR(CSET(I))
          PTR(I)=I
          IF(VAL(I).LE.0)THEN
            VAL(I)=10E-5
            RREQ(I)=RAVAIL
          ENDIF
        ENDIF
100   CONTINUE
      CALL  SORT6(VAL,RREQ,PTR,NCSET)
      RETURN
      END





C
C
C
      SUBROUTINE SETDUR(IPRNT)
C
C-----THIS SUBROUTINE TAKES A FINAL SELECTION AND ATTEMPTS
C       TO ADJUST
C-----THE ACTIVITY DURATIONS TO OBTAIN A MORE COMPLETE
C       UTILIZATION
C-----OF THE AVAILABLE RESOURCES
C
C------------------------------------------------------------
C       --------
C
C----------------------PROBLEM PARAMETERS-------------------
C       --
C
      INTEGER SPTR(250),SUCC(1000),PPTR(250),PRED(1000),
     *  DMIN(250),
     *        DMAX(250),TL(250),SAV(250),PROFIL(25,3)
      COMMON/PROB/NACT,TLAB,TVAL,NINC,TRES,SPTR,SUCC,PPTR,
     *  PRED,
     *             DMIN,DMAX,TL,SAV,PROFIL
C
C----------------------PROBLEM PARAMETERS-------------------
C       --
C----------------------KNAPSACK PARAMETERS------------------
C       --
C
      INTEGER XS(250),PTR(250)
      DIMENSION RREQ(250),VAL(250)
      COMMON/KNAP/BRHS,BOU,S,RL,RLB,BUDGE,RREQ,VAL,XS,PTR
C
```

```
C----------------------KNAPSACK PARAMETERS-----------------
C        --
C----------------------SOLUTION PARAMETERS-----------------
C        --
C
      INTEGER IFLAG(250),ASET(250),CSET(250),SSET(250),
     *  ES(250),
     *        LS(250),DUR(250),MD(250)
      INTEGER PINC,TIME,TNEXT,TDROP,TMAX
      COMMON/SOLNC/PINC,NASET,NCSET,NSSET,TIME,TNEXT,TDROP,
     *  TMAX,
     *             RASET,RAVAIL,RADROP,RUDROP,NSKED,SVAL,
     *  SLAB,
     *             NDEC,NKNAP,TKNAP,TTOT,SFAC1,SFAC2
      COMMON/SOLNA/IFLAG,ASET,CSET,SSET,ES,LS,DUR,MD
C
C----------------------SOLUTION PARAMETERS-----------------
C        --
C
C
C----------------------------------------------------------
C        --------
C----------------------------------------------------------
C        --------
C
C-----SORT SSET BY DECREASING MINIMUM DELAY
      IF(IPRNT.GT.1)WRITE(*,90000)NSSET,(SSET(I),I=1,NSSET)
90000 FORMAT(/" SETDUR,NSSET=",I5/(15I5))
      IF(IPRNT.GT.2)WRITE(*,90001)(SSET(I),DUR(SSET(I)),I=1,
     *  NSSET)
90001 FORMAT(/" DURATIONS IN"/(2I10))
      IF (NSSET.EQ.1)GO TO 100
      CALL SORT3(SSET,MD,NSSET)
      RSLK1=RAVAIL-RSSET
      RSLK2=RADROP-RUDROP
100   IHIT=0
      JJ=1
200   J=SSET(JJ)
      IF(DUR(J).EQ.DMIN(J))THEN
        JJ=JJ+1
        IF(JJ.LT.NSSET)GO TO 200
        GO TO 500
      ENDIF
      RUP=TL(J)*(-1./DUR(J)+1./(DUR(J)-1))
      IF(TIME+DUR(J).GT.TDROP)THEN
        IF(RUP.LE.MIN(RSLK1,RSLK2))THEN
          DUR(J)=DUR(J)-1
          RSLK1=RSLK1-RUP
          RSLK2=RSLK2-RUP
          IHIT=1
          ENDIF
        ELSE
          IF(RUP.LE.RSLK1)THEN
            DUR(J)=DUR(J)-1
            RLSK1=RSLK1-RUP
            IHIT=1
          ENDIF
        ENDIF
        IF(IHIT.NE.0)THEN
          IF(JJ.EQ.NSSET)GO TO 100
          J1=JJ
          TEMP=SSET(J1)
300       J2=JJ+1
          IF(MD(TEMP).LT.MD(SSET(J2)))THEN
            SSET(J1)=SSET(J2)
            IF(J2.LT.NSSET)GO TO 300
            SSET(J2)=TEMP
          ENDIF
          IHIT=0
        ELSE
          JJ=JJ+1
        ENDIF
400     IF(JJ.LE.NSSET)GO TO 200
500     IF(IHIT.NE.0)GO TO 100
510     CONTINUE
        IF(IPRNT.GT.2)WRITE(*,90002)(SSET(I),DUR(SSET(I)),I=1,
     *    NSSET)
90002 FORMAT(/" DURATIONS OUT"/(2I10))
C-----ONCE THE DURATIONS HAVE BEEN DETERMINED, FIX THEM
C-----PERMENANTLY BY MODIFYING DMIN AND DMAX
        DO 600 J=1,NSSET
          DMIN(SSET(J))=DUR(SSET(J))
          DMAX(SSET(J))=DUR(SSET(J))
600     CONTINUE
        RETURN
        END





C
C
C
        SUBROUTINE EARLY(NODE,TIME,DUR,ES,IFLAG)
C
C------SUBROUTINE TO COMPUTE EARLY START SCHEDULE FOR THE
C         RESOURCE
C-----CONSTRAINED SCHEDULING HEURISTIC.
```

```
C
C
C-----------------------------------------------------------------------
C        ---------
C
C
C------------------------PROBLEM PARAMETERS------------------
C        --
C
      INTEGER SPTR(250),SUCC(1000),PPTR(250),PRED(1000),
     *  DMIN(250),
     *        DMAX(250),TL(250),SAV(250),PROFIL(25,3)
      COMMON/PROB/NACT,TLAB,TVAL,NINC,TRES,SPTR,SUCC,PPTR,
     *  PRED,
     *            DMIN,DMAX,TL,SAV,PROFIL
C
C------------------------PROBLEM PARAMETERS------------------
C        --
C------------------------KNAPSACK PARAMETERS-----------------
C        --
C
      INTEGER XS(250),PTR(250)
      DIMENSION RREQ(250),VAL(250)
      COMMON/KNAP/BRHS,BOU,S,RL,RLB,BUDGE,RREQ,VAL,XS,PTR
C
C------------------------KNAPSACK PARAMETERS-----------------
C        --
C
      INTEGER DUR(250),ES(250),IFLAG(250),TIME,DTEMP
C-----------------------------------------------------------------------
C        ---------
C-----------------------------------------------------------------------
C        ---------
C
C-----INITIALIZE ES
      DO 100 I=NODE,NACT
        IF(IFLAG(I).LE.0)THEN
          ES(I)=TIME
        ELSE
          ES(I)=IFLAG(I)-DUR(I)
        ENDIF
100   CONTINUE
C-----CALCULATE SCHEDULE TIMES
      DO 200 I=NODE,NACT-1
        ITIME=ES(I)+DUR(I)
        IFIRST=SPTR(I)
        ILAST=SPTR(I+1)-1
        IF(IFIRST.LE.ILAST)THEN
          DO 150 K=IFIRST,ILAST
            IF(ES(SUCC(K)).LT.ITIME)THEN
```

```
              ES(SUCC(K))=ITIME
            ENDIF
150       CONTINUE
        ENDIF
200   CONTINUE
      RETURN
      END




C
C
C
       SUBROUTINE LATE(NODE,TIME,DUR,LS,IFLAG)
C
C------SUBROUTINE TO COMPUTE LATE START SCHEDULE FOR THE
C        RESOURCE
C-----CONSTRAINED SCHEDULING HEURISTIC.
C
C
C-----------------------------------------------------------
C        ---------
C
C
C----------------------PROBLEM PARAMETERS-------------------
C        --
C
      INTEGER SPTR(250),SUCC(1000),PPTR(250),PRED(1000),
     *  DMIN(250),
     *        DMAX(250),TL(250),SAV(250),PROFIL(25,3)
      COMMON/PROB/NACT,TLAB,TVAL,NINC,TRES,SPTR,SUCC,PPTR,
     *  PRED,
     *            DMIN,DMAX,TL,SAV,PROFIL
C
C----------------------PROBLEM PARAMETERS-------------------
C        --
C----------------------KNAPSACK PARAMETERS------------------
C        --
C
      INTEGER XS(250),PTR(250)
      DIMENSION RREQ(250),VAL(250)
      COMMON/KNAP/BRHS,BOU,S,RL,RLB,BUDGE,RREQ,VAL,XS,PTR
C
C----------------------KNAPSACK PARAMETERS------------------
C        --
C
      INTEGER DUR(250),LS(250),IFLAG(250),TIME,DTEMP
```

```
C------------------------------------------------------------
C       ---------
C------------------------------------------------------------
C       ---------
C
C-----INITIALIZE LS
      DO 100 I=NODE,NACT
        IF(IFLAG(I).LE.0)THEN
          LS(I)=TIME
        ELSE
          LS(I)=IFLAG(I)-DUR(I)
        ENDIF
100   CONTINUE
C-----CALCULATE SCHEDULE TIMES
      DO 200 I=NACT,NODE,-1
        ITIME=LS(I)
        IFIRST=PPTR(I)
        ILAST=PPTR(I+1)-1
        IF(IFIRST.LE.ILAST)THEN
          DO 150 K=IFIRST,ILAST
          DTEMP=DUR(PRED(K))
            IF(LS(PRED(K)).GT.ITIME-DTEMP)LS(PRED(K))=ITIME-
     *  DTEMP
150       CONTINUE
        ENDIF
200   CONTINUE
      RETURN
      END





C
C
C
      SUBROUTINE SORT1(SET,NSET)
C
C-----SORT ROUTINE TO ORDER "SET" BY INCREASING VALUE OF
C-----ELEMENTS OF "SET"."NSET" IS THE SIZE OF "SET"
C
      INTEGER SET(1)
C
C
      IF(NSET.EQ.1)RETURN
      K=NSET
      FLAG=NSET
100   IF(FLAG.EQ.0)RETURN
      K=FLAG-1
      FLAG=0
      DO 200 J=1,K
      IF(SET(J).GT.SET(J+1))THEN
         ITEMP=SET(J)
         SET(J)=SET(J+1)
         SET(J+1)=ITEMP
         FLAG=J
       ENDIF
200   CONTINUE
      GO TO 100
      END







C
C
      SUBROUTINE SORT3(SET,VAL,NSET)
C
C-----SORT ROUTINE TO ORDER "SET" BY INCREASING VALUE OF
C-----ELEMENTS OF "VAL"."SET" POINTS INTO "VAL".
C
      INTEGER SET(250)
      DIMENSION VAL(250)
C
C
      IF(NSET.EQ.1)RETURN
      K=NSET
      FLAG=NSET
100   IF(FLAG.EQ.0)RETURN
      K=FLAG-1
      FLAG=0
      DO 200 J=1,K
      IF(VAL(SET(J)).GT.VAL(SET(J+1)))THEN
         ITEMP=SET(J)
         SET(J)=SET(J+1)
         SET(J+1)=ITEMP
         FLAG=J
       ENDIF
200   CONTINUE
      GO TO 100
      END
      SUBROUTINE PROBIN(IPRNT)
C
C-----------------------------------------------------------------
C        --
C
C----------------------PROBLEM PARAMETERS-------------------------
C        --
C
      INTEGER SPTR(250),SUCC(1000),PPTR(250),PRED(1000),
     *  DMIN(250),
     *        DMAX(250),TL(250),SAV(250),PROFIL(25,3),
     *  NAME(250)
      COMMON/PROB/NACT,TLAB,TVAL,NINC,TRES,SPTR,SUCC,PPTR,
     *  PRED,
     *            DMIN,DMAX,TL,SAV,PROFIL,NAME
C
C----------------------PROBLEM PARAMETERS-------------------------
C        --
C
      INTEGER UNIT1,UNIT2,UNIT3,UNIT4
      COMMON/PARMS/UNIT1,UNIT2,UNIT3,UNIT4
C-----------------------------------------------------------------
C        --
C
      READ(UNIT1,*)NACT,NPRED,IDUM,TLAB,TVAL
      DO 100 I=1,NACT
        READ(UNIT1,*)NAME(I),PPTR(I),SPTR(I),TL(I),SAV(I),
     *               DMIN(I),DMAX(I)
100   CONTINUE
      READ(UNIT1,*)(PRED(K),K=1,NPRED)
      READ(UNIT1,*)(SUCC(K),K=1,NPRED)
      SPTR(NACT+1)=NPRED+1
      PPTR(NACT+1)=NPRED+1
      IF(IPRNT.GT.3)THEN
        DO 150 I=1,NACT
          I1=PPTR(I)
          I2=PPTR(I+1)-1
          I3=SPTR(I)
          I4=SPTR(I+1)-1
          WRITE(*,90000)I,(PRED(K),K=I1,I2)
          WRITE(*,90000)I,(SUCC(K),K=I3,I4)
150    CONTINUE
      ENDIF
90000 FORMAT(/" ACTIVITY ",I5/(15I5))
      SPTR(NACT+1)=NPRED+1
      PPTR(NACT+1)=NPRED+1
C
      DO 200 I=1,NACT
        IFIRST=SPTR(I)
        ILAST=SPTR(I+1)-1
```

```
          IF(IFIRST.LE.ILAST)THEN
            DO 175 K=IFIRST,ILAST
              IF(SUCC(K).LT.I)CALL ERRSTP(400,UNIT4)
175         CONTINUE
          ENDIF
200     CONTINUE
        RETURN
        END
```

APPENDIX C

TEST PROBLEM GENERATOR LISTING

```
      PROGRAM TSTS(INPUT,OUTPUT,TAPE2,TAPE3)
C...  THIS PROGRAM GENERATES TEST PROBLEMS FOR THE SHIPYARD
C        PROJECT. THE
C...  LINKED LISTS ARE USED TO STORE DATA.
C...  AND TAPE 2&3 REPRESENT:
C...         TAPE2 = ACTIVITY FILE (NAME, TOTAL LABOR,
C       DURATION INTERVALS,
C...                 L SAVING,POINTERS INTO PRECEDENCE AND
C       CONSTRAINT FILE
C...         TAPE3 = PRECEDENCE FILE (NO AND THE LIST OF THE
C       SUCCESSORS
C...                 AND PREDECESSORS)
C...  THE VARIABLES ARE DEFINED AS:
C...        PRE1(.)    = ARRAYS USED TO STORE THE LINKED LISTS
C       OF DATA DEFIN
C...        PRE2(.)      THE PREDECESSORS OF EACH ACTIVITY (
C       DIMENSION =
C...                     N + TOTAL NO. OF ARCS )
C...        SUC1(.)    = ARRAYS USED TO STORE THE LINKED LISTS
C       OF DATA DEFIN
C...        SUC2(.)      THE SUCCESSORS OF EACH ACTIVITY
C       (DIMENSION =
C...                     N + TOTAL NO. OF ARCS )
C...        NE              = NO OF ELEMENTS
C...        NU              =  "  " UNITS
C...        PERC1           = MULTIPLIER FOR THE NO OF RANDOM
C       ARCS FOR
C...                          ELEMENTS AND UNITS.
C...        PERC2           = MULTIPLIER FOR THE NO OF RANDOM
C...                          BETWEEN ELEMENTS AND UNITS
C...        PTRP            = POINTER TO THE PRECEDENCE  FILE
C
C

      INTEGER PRE1(2000),PRE2(2000),SUC1(2000),ARCS(150,150)
      INTEGER SUC2(2000),BLK1,BLK2,PTRP
      COMMON/A/PRE1,PRE2/B/SUC1,SUC2,KP,KS
      DATA PERC1,PERC2,DEB1,DEB2,DUB1,DUB2/0.9,0.1,7.,9.,
     +   10.,12./
      DATA DEBR1,DEBR2,DUBR1,DUBR2/12.,14.,16.,18./
      DATA CREB1,CREB2,CRUB1,CRUB2,ISTD/8.33,9.,8.33,9.,2/
      DATA CREBR1,CREBR2,CRUBR1,CRUBR2/9.38,10.18,9.38,
     +   10.18/
      SEED=FLOAT(723+8  .AND.77777777777B)
      CALL RANSET(SEED)
      PRINT*"INPUT NO OF ELEMENTS , NO OF UNITS"
      READ*,NE,NU
      PRINT*"DEFAULT PARAMETERS ? (1=YES, 0=NO)"
      READ*,IY
      IF(IY.EQ.1)GO TO 5
      PRINT*"INPUT MULTIPLIER FOR NO OF RANDOM ARCS"
      PRINT*"FOR ELEMENT AND UNITS ON BLOCK"
      READ*,PERC1
      PRINT*"INPUT MULTIPLIER FOR NO OF RANDOM ARCS"
      PRINT*"BETWEEN ELEMENTS AND UNITS ON BLOCK"
      READ*,PERC2
      PRINT*"DO YOU WANT TO CHANGE MIN AND MAX DURATIONS
     *  AND"
      PRINT*"THE RANGE OF CREW SIZE ? (1=YES, 0=NO)"
      READ*,IY
      IF(IY.EQ.0)GO TO 5
      PRINT*"INPUT MIN DUR., MAX DUR., LOWER AND UPPER
     *  RANGE ",
     *"OF CREW SIZE"
      PRINT*"E.G.  7,9,8.33,9"
      PRINT*"FOR ELEMENTS ON BLOCK"
      READ*,DEB1,DEB2,CREB1,CREB2
      PRINT*"FOR UNITS ON BLOCK"
      READ*,DUB1,DUB2,CRUB1,CRUB2
      PRINT*"FOR ELEMENTS ON BOARD"
      READ*,DEBR1,DEBR2,CREBR1,CREBR2
      PRINT*"FOR UNITS ON BOARD"
      READ*,DUBR1,DUBR2,CRUBR1,CRUBR2
C  INITIALIZATION
C
5     ISIZE=2000
      DO 10 I=1,ISIZE
      PRE1(I)=0
      PRE2(I)=0
      SUC1(I)=0
      SUC2(I)=0
10    CONTINUE
C  CALCULATE THE NO. OF RANDOM ARCS
      NOA1=FLOAT(NE)*PERC1+0.5
      NOA1M=(((NE-1)*NE)/2)*0.6
      IF(NOA1.GT.NOA1M)NOA1=NOA1M
      NOA2=FLOAT(NU)*PERC1+0.5
      NOA2M=(((NU-1)*NU)/2)*0.8
      IF(NOA2.GT.NOA2M)NOA2=NOA2M
      NOA3=(FLOAT(NE)+FLOAT(NU))*PERC2+.5
      NOA3M=((NE*NU)/4)*0.8
      IF(NOA3.GT.NOA3M)NOA3=NOA3M
C
C  COMPUTE THE STARTING AND FINISHING NUMBERS FOR NODES
C     DEFINING
C   ON-BLOCK
      BLK1=NE+NU+1
      BLK2=BLK1+NE+NU-1
      KP=KS=BLK2+1
C  GENERATE RANDOM ARCS FOR ELEMENTS ON BLOCK
```

```
      Z1=BLK1
      Z2=BLK1+N1-1
      NCNT=0
20    N1=BLK1+(Z2-Z1)*RANF(SEED)+0.5
      IF((Z2-N1).LT.1.0)GO TO 20
21    N2=N1+(Z2-N1+1)*RANF(SEED)
      IF(N2.EQ.N1.OR.N2.GT.Z2)GO TO 21
      IF(ARCS(N1,N2).NE.0)GO TO 20
      ARCS(N1,N2)=1
      CALL LINK(N1,N2)
      NCNT=NCNT+1
      IF(NCNT.LT.NOA1)GO TO 20
      NCNT=0
      Z3=Z2+1
      Z4=BLK2
30    N1=Z3+(Z4-Z3)*RANF(SEED)
      IF((Z4-N1).LT.1.0)GO TO 30
31    N2=N1+(Z4-N1+1)*RANF(SEED)
      IF(N2.EQ.N1.OR.N2.GT.Z4)GO TO 31
      IF(ARCS(N1,N2).NE.0)GO TO 30
      ARCS(N1,N2)=1
      CALL LINK(N1,N2)
      NCNT=NCNT+1
      IF(NCNT.LT.NOA2)GO TO 30
      NCNT=0
      Z12=(Z1+Z2)/2.
      Z34=(Z3+Z4)/2.
35    IF(RANF(SEED).LT.0.5)GO TO 40
36    N1=Z1+(Z12   -Z1)*RANF(SEED)
      N2=Z34+(Z4-Z34)*RANF(SEED)
      IF(ARCS(N1,N2).NE.0)GO TO 36
      GO TO 41
40    N1=Z3+(Z34-Z3)*RANF(SEED)
      N2=Z12+(Z2-Z12)*RANF(SEED)
      IF(ARCS(N1,N2).NE.0)GO TO 40
41    ARCS(N1,N2)=1
      CALL LINK(N1,N2)
      NCNT=NCNT+1
      IF(NCNT.LT.NOA3)GO TO 35
C
C  PRINT THE OUTPUT
C
      PTRP=-1
      KOUNT=BLK1+N1-1
      WRITE(2,*)N1,NU
      DO 100 I=BLK1,BLK2
C
C  MAKE THE ACTIVITY FILE
      WRITE(2,*)I
C  CALCULATE AVG. DURATION FOR ACTIVITIES ON BLOCK
```

```
      IF(I.LE.KOUNT)GO TO 55
      DEB1=DUB1
      DEB2=DUB2
      CREB1=CRUB1
      CREB2=CRUB2
 55   IAVGD=DEB1+(DEB2-DEB1+1.)*RANF(SEED)
      AVGD=IAVGD
      CREW=CREB1+(CREB2-CREB1)*RANF(SEED)
      LBR=CREW*AVGD*8.+.5
      WRITE(2,*)LBR
      WRITE(2,*)(IAVGD-ISTD),(IAVGD+ISTD)
C  CALCULATE   SAVING FOR ON BLOCK ACTIVITIES
C  FIRST GENERATE TOTAL LABOR FOR CORRESPONDING ON-BOARD
C        ACTIVITY
      IF(I.LE.KOUNT)GO TO 57
      DEBR1=DUBR1
      DEBR2=DUBR2
      CREBR1=CRUBR1
      CREBR2=CRUBR2
 57   IAVGD=DEBR1+(DEBR2-DEBR1+1.)*RANF(SEED)
      AVGD=IAVGD
      CREW=CREBR1+(CREBR2-CREBR1)*RANF(SEED)
      LBR2=CREW*AVGD*8.+.5
      FACTOR=.5+RANF(SEED)
      SAVING=(FLOAT(LBR2)-FLOAT(LBR))*15.*FACTOR
      WRITE(2,*)SAVING
      PTRP=PTRP+2
      WRITE(2,*)PTRP
C  MAKE THE PRECEDENCE FILE
      WRITE(3,*)SUC1(I)
      WRITE(3,*)PRE1(I)
      IF(SUC1(I).EQ.0)GO TO 65
C  WRITE SUCCESSORS
      J=SUC2(I)
 60   JJ=SUC1(J)
      WRITE(3,*)JJ
      PTRP=PTRP+1
 62   J=SUC2(J)
      IF(J.NE.0)GO TO 60
 65   IF(PRE1(I).EQ.0)GO TO 100
C  WRITE PREDECESSORS
      J=PRE2(I)
 67   JJ=PRE1(J)
      WRITE(3,*)JJ
      PTRP=PTRP+1
 70   J=PRE2(J)
      IF(J.NE.0)GO TO 67
 100  CONTINUE
      PRINT*,"NUMBER OF ACTIVITIES= ",NL+NU
      PRINT*,"NUMBER OF ARCS= ",NOA1,NOA2,NOA3
      END




C           ****************************************
      SUBROUTINE LINK(J,JJ)
C           ****************************************
      INTEGER PRE1(2000),PRE2(2000),SUC1(2000),SUC2(2000)
      COMMON/A/PRE1,PRE2/B/SUC1, SUC2,KP,KS
C  MAKE THE LINKED LISTS FOR SUCCESSORS
      SUC1(KS)=JJ
      SUC2(KS)=SUC2(J)
      SUC2(J)=KS
      KS=KS+1
      SUC1(J)=SUC1(J)+1
C  MAKE THE LINKED LISTS FOR PREDECESSORS
      PRE1(KP)=J
      PRE2(KP)=PRE2(JJ)
      PRE2(JJ)=KP
      KP=KP+1
      PRE1(JJ)=PRE1(JJ)+1
      RETURN
      END
```