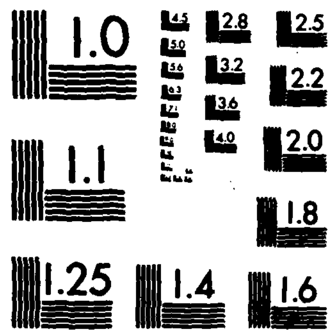
AD-A141 198 ENHANCEMENTS TO THE NETWORK REPAIR LEVEL ANALYSIS (NRLA) MODEL USING MARG..(U) AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOOL OF ENGI..

1/2

UNCLASSIFIED G W ARNETT ET AL. DEC 83 AFIT/GOR/OS/83D-3 F/G 15/5 NL


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
5.6

AFIT/GOR/OS/83D-3

AD-A141 198

ENHANCEMENTS TO THE NETWORK REPAIR LEVEL ANALYSIS (NRLA) MODEL USING MARGINAL ANALYSIS TECHNIQUES AND CENTRALIZED INTERMEDIATE REPAIR FACILITY (CIRF) MAINTENANCE CONCEPTS

THESIS

Gary W. Arnett, Captain, USAF
Nicholas W. Reybrock, Captain, USAF

AFIT/GOR/OS/83D-3

84 05 15 034

AFIT/GOR/OS/83D-3

ENHANCEMENTS TO THE NETWORK REPAIR LEVEL ANALYSIS (NRLA) MODEL USING MARGINAL ANALYSIS TECHNIQUES AND CENTRALIZED INTERMEDIATE REPAIR FACILITY (CIRF) MAINTENANCE CONCEPTS

THESIS

Presented to the Faculty of the School of Engineering

of the Air Force Institute of Technology

Air University

In Partial Fulfillment of the

Requirements for the Degree of

Master of Science in Operations Research

Gary W. Arnett

Captain, USAF

Nicholas W. Reybrock

Captain, USAF

December 1983

Approved for public release; distribution unlimited

Acknowledgements

This research effort has been conducted under the direct sponsorship of the Concepts and Analysis Division, Air Force Acquisition Logistics Center (XRS/AFLC) Wright Patterson Air Force Base, Ohio. Mr. Larry Brisken provided us the initial guidance and understanding of the NRLA model, which helped us define our objectives and develop a thesis strategy.

Mr. Terry Mitchell's perceptive insights and detailed knowledge of the program's logic have been of immeasurable value to the successful accomplishment of our intended objectives. We also especially express our appreciation and thanks to our advisor, Lt Col Gerald R. Armstrong, and our reader, Lt Col Tom Clark, Jr., for their professional guidance and technical expertise.

Finally, we take this opportunity to thank our wives, Kay and Michelle, and our children, Becky, Brian, and Kasey, whose support and help made the completion of this product a reality.

Gary W. Arnett

Nicholas W. Reybrock

Table of Contents

List of Figures

List of Tables

Abstract

Applied network theory and marginal analysis concepts were utilized to design, computerize, verify, and evaluate, three major software modifications to the Network Repair Level Analysis (NRLA) model. First, a preprocessor subroutine using marginal analysis techniques was developed and tested to reduce the computer processing requirements of the program. Second, a new network labeling algorithm which solve the max-flow min-cut problem is presented. This algorithm performs 100 times faster than the current algorithm and 73 times faster than the highly efficient, commercially available, primal networking code known as GNET. Third, for the first time a networking structure has been designed which allows for the inclusion of Centralized Intermediate Repair Facilities (CIRF) in the repair level analysis decision process.

These products greatly expand the NRLA model's capability while at the same time improving its operational efficiency. Through their integration and use, System Program Managers have a comprehensive analytical tool to effectively conduct repair level analysis and to design more cost-effective logisitical structures to support the operation of Air Force systems.

The NRLA program is hosted on the CREATE Operating System and contains approxiamately 5500 lines of computer code. It consists of a main routine and twelve major subroutines. The

results from the NRLA model are used by logistical planners to quantify the potential cost impacts associated with alternative maintenance plans. As the technological complexity of weapons systems has increased new and innovative logisitcal support systems are required to maximize the system's operational capability while minimizing life cycle costs. The above enhancements to the NRLA model were designed to meet these new challenges. This research effort was sponsored by the Concepts and Anaylsis Division, Air Force Acquisition Logistics Center (XRS/AFALC/AFLC).

I. Problem Definition

Introduction

Every program manager, who is responsible for acquiring a new weapon system to be used by tomorrow's United States Air Force (USAF), faces a common challenge. It consists of three, sometimes conflicting, functional goals; system performance, system cost, and acquisition schedule. His goal is a formidable one. He must attempt to maximize the performance of the system at the least possible cost to the government while maintaining the scheduled time-frame for procurement of the system. A primary ingredient in the successful accomplishment of this effort is an accurate assessment of the "operational support requirements and limitations of the system" (13:3) and the timely consideration of alternative maintenance concepts capable of achieving a desired level of system effectiveness.

During the Vietnam conflict, the importance of developing a valid maintenance plan as an integral part of the system's engineering development became brutally apparent. Many times new systems were deployed to meet the changing threat only to have their components subsequently fail causing excessive downtime for the system. An example of this was the AN/TRC-87 UHF radio set. The design of this radio set did not anticipate its extensive use in a jungle environment, as a result major component failures occured which required depot repair (1:134). The availability, dependability, and capability of these systems were all substantially reduced because of a lack of adequate attention being given to their reliability, the resources necessary to support their repair, and specific identification of the repair location for their components. As a direct result of these experiences, the Department of Defense (DOD) community and the Air Force have taken great strides to ensure that such decisions become an integral part of the system's design. The Integrated Logistic Support (ILS) Program, established in 1972 by AFR 800-8, requires that planning for cost-effective logistic support, including analysis of repair-level alternatives, be conducted during the acquisition phase of a weapon system (11:3). In conjunction with this guidance, "AFSC/AFLCR 800-28 establishes Air Force policies and procedures with respect to Repair Level Analysis (RLA)" (14:4). These policies and procedures are designed to ensure that alternative maintenance concepts are considered during conceptual development and at the appropriate times in the life cycle of the system. Additionally, AFSC/AFLC Regulation 800-28 identifies RLA as a separate evaluation factor during the source selection process. This is a further indication of the emphasis and recognition that is now being given repair level decision making (13:3).

The Repair Level Analysis process is the procedure by which economic comparisons are made between repair locations to develop a comprehensive and cost effective maintenance plan. RLA encompasses a wide variety of analytical techniques and methods. These techniques can be used separately or in combination to provide the system program offices with economically based management information as to the "best" set of repair level decisions for their system. In an effort to document the strengths and weaknesses of these different methods and to assist logistics analysts in determining the appropriate technique for a particular application, the Air Force has recently updated these procedures in AFLC/AFSC Pamphlet 800-4, Repair Level Analysis Procedures (revised June 1983). Formerly known as Optimum Repair Level Analysis (ORLA) (12), RLA now encompasses the following methods (13:5):

(1) Network Repair Level Analysis (NRLA)
(2) Item Repair Level Analysis (IRLA)
(3) Marginal Analysis Repair Level Analysis (MARLA)
(4) Equal Cost Curves (ECC)
(5) SE/Pipeline Ratio

Each of these techniques are clearly explained in AFLCP/AFSCP 800-4. However, the NRLA conceptual model, which were introduced in 1980, are recognized as the most comprehensive approach to repair level analysis and has consequently received the widest use and attention (13:5). Since NRLA's implementation, a question has arisen as to its ability to continue to provide reliable and consistent repair level logistics cost effectiveness and the operational capability of future systems. This research has been initiated in an effort to answer this question. The remainder of this chapter will deal with the scope and limitations of the NRLA model with an emphasis on identifying software enhancements to correct identified shortcomings.

Background

Prior to 1980 repair level analysis for Air Force systems was conducted on an item by item basis using a method called Optimum Repair Level Analysis (ORLA) (12:1). In ORLA each Line Replaceable Unit (LRU) and Shop Replaceable Unit (SRU) in the system would be separately priced for each of three repair options; (1) repair at intermediate base, (2) repair at depot and (3) discard or scrap. The repair option associated with the minimum cost would be selected for that particular LRU or SRU. The problem with this method of determining repair level decisions was that the SE costs associated with each LRU/SRU repair were prorated and estimated according to that LRU/SRUs use of the SE resource. Thus a LRU/SRU which required 20 percent of the SE's available repair time would need to economically justify at least 20 percent of that SE cost. In addition if all LRU/SRUs requiring that SE were not chosen for the same repair level then the percent utilization and prorated costs would need to be recomputed based on the new set of LRU/SRUs at that level. This could then lead to other LRU/SRUs not being able to support their required SE costs which would lead to further proration of costs and possibly to all LRU/SRUs being repaired at the depot level when in fact the total set of SE related LRU/SRUs could easily justify base level repair as the most economical option (8:6-7). A second problem with the ORLA method of determining repair level decisions was that the Air force had no official computer program to implement ORLA. This resulted in virtual chaos when evaluating contractor estimates, in that contractors did not use standardized software to conduct their analysis.

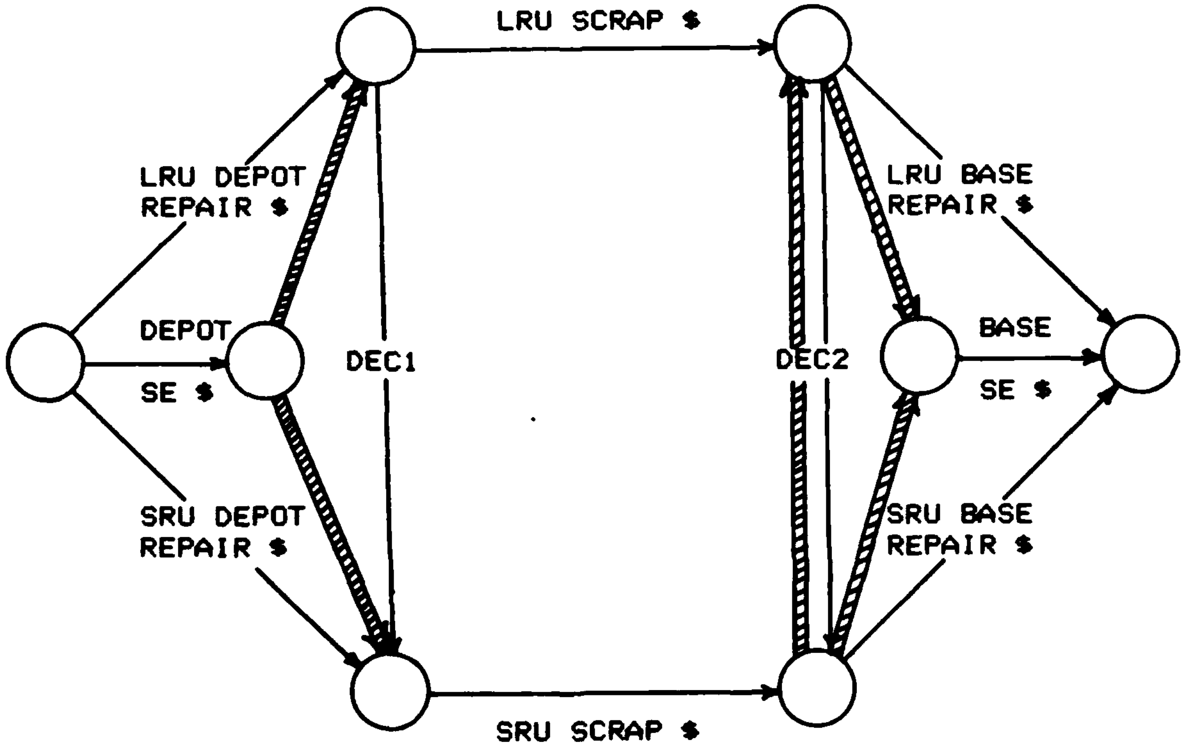
To solve these problems the NRLA program was developed by the Concepts and Analysis division, XRS/AFALC(AFLC) giving the Air force, for the first time, the capability to use a systems approach in the development and formulation of repair level decisions. NRLA was developed as a FORTRAN based software package that is transportable between computer systems and is therefore useable by government contractors providing a consistent framework for evaluation of their proposals.

NRLA

The formulation of the repair level analysis problem as a network provides a couple of advantages over previous RLA methodologies. It takes into account the LRU and SRU indenture relationships which keeps it from making inconsistent decisions. An example of this would be deciding to scrap a LRU but repair its indentured SRUs. The model also treats each piece of support equipment as a common resource which is shared by a group of LRUs and/or SRUs. The repair level decisions are then made, based on these LRU/SRU group's ability to economically support the purchase of the support equipment for a particular repair level. All of the repair decisions are determined simultaneously for all of the failure modes of a group of LRUs and their associated SRUs, thus the decisions that are made are optimal for the entire group of items.

The actual formulation of the network is shown in Figure 1. The nodes of the network represent the LRU failure modes, the SRUs, and the SE resources required for the LRU/SRU repairs. The arcs of the network represent the costs associated with the different repair level options. Table I defines the specific arc costs for the network in terms of eleven types of logistical costs. The NRLA Users Guide available from XRS/AFALC(AFLC) details the computation of these costs in terms of the LRU/SRU's cost, MTBF, and standard maintenance and supply factors. The heavy arcs are dummy arcs which provide the LRU/SRU/SE interdependency relationships. The capacities of the dummy arcs are set such that they will never enter into the solution set.

The repair level decisions are obtained by applying an optimization algorithm to the network to solve the maximin flow minimum cut problem. This algorithm provides a network solution with a cut set that describes the unique optimum set of minimum cost repair level decisions for the entire system. It should be pointed out that the NRLA model in no way attempts to compute a total life cycle cost for a particular set of repair decisions but, "includes only those costs which directly impact the repair level decision". Examples of costs which are not included are: repair in place costs, and the costs associated with removing the failed LRU/SRU from the end item (8:3).



NOTE: DEC1 and DEC2 are SRU costs which are incurred if the LRU decision is base level repair and (a) the SRU decision is depot repair (DEC1), or (b) the SRU decision is either depot repair or scrap (DEC2).

FIGURE 1. Basic Structure of an RLA Network

TABLE I

Relationship Between Logistics and Network Costs

	NETWORK DECISION FACTORS									
	LRU REPAIR			SRU REPAIR					SE	
	DEPOT	SCRAP	BASE	DEPOT	SCRAP	BASE	DEC1	DEC2	DEPOT	BASE
LOGISTIC FACTORS										
1. Support Equipment									X	X
(1a) Acquisisition									x	x
(1b) Ops & Maint									x	x
(1c) Facilities									x	x
2. Tech. Data Acquis.	X		X	X		X				
3. Maint Training	X		X	X		X				
4. Repair Labor	X		X	X		X				
5. Item Entry	X		X	X		X				
6. Supply Admin.			X			X				
7. Repair Material	X		X	X		X				
8. Packing & Shipping	X	X	X			X	X	X		
9. Base Spares Quantity	X	X	X			X		X		
10. Depot Spares Quantity	X						X			
11. Replacement Spares		X			X					

Assumptions

Virtually any model makes some simplifying assumptions which enables it to calculate a solution or make some determination/conclusion about the system being studied. The NRLA model is no exception to this rule. The following assumptions have been identified as relevant to the solutions derived by NRLA (8:3-6).

(1) The user specifies the number of bases and the number of end items per base (assumed to be equal for all bases). All depot repair of a particular LRU/SRU is accomplished at the same depot location.

(2) Base level maintenance system data are equal for all bases and all types of repair tasks. The corresponding depot data factors are constant also.

(3) Supply system data factors are constant for all LRUs and SRUs analyzed. ie. shipping times from depot to any CONUS location are equal, and the same is true for shipments from depot to any overseas location.

(4) Only a single set of technical data is purchased from the contractor.

(5) Preventative and scheduled maintenance actions are not addressed by the model.

(6) The model explicitly evaluates each LRU failure mode for a repair level decision, however SRU failure modes are assumed to be similar enough to allow for considering them all equal.

(7) Maintenance man-hours for repair and SE utilization are assumed to be equal.

(8) The depot stock level of SRUs is designed to satisfy base level demands. The stock level supports base level SRU remove and replace maintenance actions but not similar depot actions.

Limitations

The NRLA program as it exists today has proven to be a valuable tool in such areas as determining optimal repair levels in support of the provisioning process, identification and justification of LRU/SRU support equipment requirements, and assessment of a system's sensitivity to cost and reliability growth. However, limitations in the programs efficiency and capability have resulted in the need to enhance and in some cases expand the software of the program to accommodate larger and more complex systems and to be able to evaluate new maintenance concepts.

Approximately 50-60 contractors and system program offices are currently using the NRLA model as implemented in 1980. User comments indicate that approximately 60% of the analyses that they perform using NRLA are conducted on a piecemeal basis with subsystems being analyzed individually. This is due to the fact that it is easier and more convenient to analyze and work with sub-systems. However, the main reason which drives users to analyze smaller sections of the system is computer resources. As the system being analyzed gets larger and more complex it requires more computer storage space to load the program and more computer time to run it. Both storage requirements and run-time requirements are used to prioritize computer jobs with the result that as the job gets bigger the turn around time on the analysis gets longer, an undesireable result. Also once the analysis of the subsystems is complete these results are then manually cross-referenced to determine the final systems set of repair level decisions.

A second limitation in the NRLA model results from the evolution of a new maintenance concept, Centralized Intermediate Repair Facilities (CIRF). This concept is a result of the higher LRU/SRU/SE costs associated with new systems. In many instances the SE costs are too high to allow for base level repair and at the same time the pipeline costs associated with depot repair are also much higher than is desired. CIRF provides for a single central repair facility servicing several bases thereby eliminating some of the costs associated with each of the above repair options. The NRLA model currently does not allow for consideration of a CIRF maintenance concept in its optimum repair level analysis.

Thesis Objectives

The main objectives of the thesis effort deal with developing enhancements to the NRLA model which will provide solutions to the shortcomings as identified by the users; ie

(1) The excessive computer time necessary to execute the program for large systems

(2) Inability of the program to consider the option of a Centralized Intermediate Repair Facility (CIRF)

The objectives fall into two distinct areas; (1) enhancements which will allow the NRLA model to analyze larger and more complex systems and (2) enhancements which will allow the NRLA model to evaluate additional maintenance concepts, primarily CIRF. The first objective area can be further broken down into two areas (1) enhancements which increase the efficiency of the algorithm which solves the max-flow min-cut problem and (2) enhancements which reduce the size of the problem input to the NRLA model. A short description of each of these objective areas and the proposed solution methodology follows. A comprehensive treatment of these areas is contained in Chapter III.

The computational time requirements for the solution of the repair level problem by the NRLA model are driven by two major factors, one internal to the program and one external. The internal factor deals with the computational efficiency of the labelling algorithm subroutine used to solve the max-flow min-cut problem. Labeling in the context of network flows is the means by which the algorithm identifies which nodes in the network it has visited. The labels prevent the algorithm from revisiting a node thus ensuring that the flow augmentation path will be identified in a finite number of steps. The external factor affecting the computation time is the size of the system to be analyzed in terms of LRUs, SRUs, and SE. Each of these areas is addressed in the thesis effort.

The current labeling algorithm utilized in the NRLA model is one developed by Ford and Fulkerson in the early 1960s (17). The approach taken is to perform a breadth first search with labeling as you proceed to the network sink. Once the sink is reached the flow is augmented on the path and the labels erased and the process started over. In a large system the majority of the processing time is used performing the labeling process which can be very inefficient. The approach taken to reduce this processing time is to use a depth first search, as conceptually described by Horowitz and Sahni (19:268), for flow augmentation until the maximum flow is found. This is then followed by a breadth first search to provide the correct labels for the identification of the unique minimum cost set of repair level decisions.

In addition to the depth first approach for increasing the algorithm efficiency a second method of solving the problem using a large scale commercially available algorithm called GNET will be explored. This method is expected to provide a viable alternative for the solution of systems of extreme size, such as the B1 bomber and the MX missile The GNET algorithm is a primal algorithm which has been used to solve extremely large transportation and transshipment problems (7). The NRLA program will be modified to include GNET as a subroutine to solve the max-flow problem, this will once again be followed by the Ford Fulkerson labeling process to identify the unique min cost solution needed by NRLA.

Creating a mechanism to control the second factor, that of system size, was initially much more difficult to conceptualize. Normally, the configuration of a proposed weapon system does not dramatically change once the preliminary design review (PDR) between the SPO and the contractor is conducted. Therefore the types and number of items (LRUs, SRUs,etc.) which make-up the total system are given as fixed inputs to the program. However, a possible solution to this dilemma is the use of another RLA technique called Marginal Repair Level Analysis (MARLA). Simply stated, the approach is to reduce the number of items which have to be analyzed by NRLA. This will be accomplished by incorporating a marginal repair level analysis subroutine into the existing program. This software package, acting as a NRLA preprocessor, would calculate " marginal values for reparables, and then determine, based on those marginal values, whether an item should be scrapped, depot repaired, intermediate repaired "(13:67) or CIRF repaired.

The final aspect of NRLA which will be investigated is its ability to analyze a CIRF designed maintenance plan. The structure of the NRLA model is based on the traditional three-level maintenance concept (base, intermediate, depot) as detailed in AFR 60-5. Recently, an alternative maintenance approach known as Centralized Intermediate Repair Faciltles (CIRF) has emerged as an efficient way to effectively support the repair of a weapons system. Normally each base has its own dedicated intermediate repair shop, which diagnose, repair, and replace system LRUs and SRUs. On the other hand, the CIRF concept utilizes a centrally located intermediate repair facility to service multiple bases (Figure 2). In order to ensure that a comprehensive economic analysis is performed when determining the optimal set of repair level decisions an expansion and modification of the program is necessary to evaluate the cost related impacts of using a

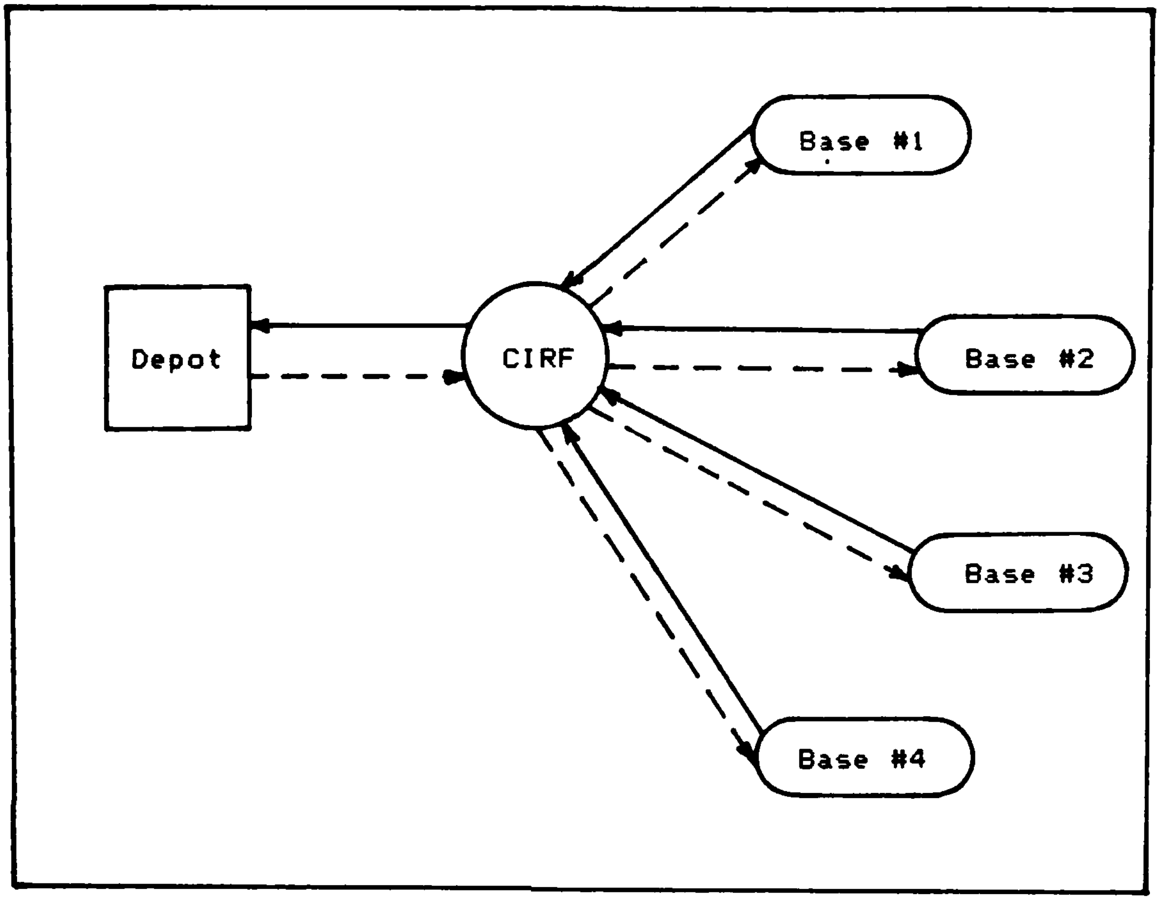


Figure 2. Centralized Intermediate Repair Facility Concept

CIRF maintenance approach. This will be the second of the two primary objectives of this thesis effort, the development of a network structure capable of analyzing a CIRF repair option. If possible this capability will be incorporated into the existing NRLA software program.

Organization

Chapter II encompasses a survey of the current literature available relating to networking algorithms with special emphasis on those articles which formulate the repair level decision based on a network structure. The thesis methodology is presented in Chapter III with a detailed discussion of the structural model developed to accomplish the stated thesis objectives. An in-depth discussion of each of the three enhancements to the NRLA model are contained in Chapters IV thru VI. Finally Chapter VII provides conclusions which can be drawn from this study as well as recommendations concerning future applications of the NRLA model.

II. Literature Review

Introduction

The real-world nature of the thesis subject directed the research effort into two principle areas: first from a practical viewpoint, a study and understanding of current USAF policies and procedures regarding Repair Level Analysis (RLA) was necessary and second, from a theoretical perspective the investigation of alternative state-of-the-art optimization techniques which could be used to solve the multi-item multi-echelon repair level decision problem was required. It was essential that each of these subjects be addressed to ensure that products generated from this effort be not only acceptable in terms of their technical content and structure, but also be compatible with existing current USAF logistics policy and doctrine.

Repair Level Analysis - A Macro View

In an effort to assist Air Force acquisition managers in the successful design and implementation of efficient maintenance structures, the Department of the Air Force has established specific guidelines and procedures to be followed by both the government and contractors in the design, analysis, and operation of the RLA program (14:1). As stated in AFSC/AFLC Regulation 800-28, Repair Level Analysis (RLA) Program, " the RLA program has two major objectives:

(1) Design-oriented RLA. This is the preliminary analysis that begins in the conceptual phase of the program and continues through the critical design review. Its goal is to evolve a design that considers the economics of support alternatives and produces an economical life cycle cost profile.

(2) Provisioning-oriented RLA.This analysis begins during the full-scale development (FSD) phase and continues into the production program. The objectives of this phase of RLA are to assign the maintenance portion of the source, maintenance, recoverability (SMR) codes and complete maintenance planning during the provisioning process."(14:1)

Both of these portions of the RLA program are conducted using mathematical models and techniques to determine the most appropriate economic level of repair for system components. However, as can be seen from the causal diagram in Figure 3, there are a multitude of factors which affect the operation of a logistics structure. This is not only reflected in terms of its costs, but also in terms of its requirements for; specialized equipment, manpower with certain maintenance skills, and dedicated facilities at the repair site. Unfortunately, individual identification and assessment of these factors on a system's operational effectiveness is only an initial step in the process of developing a total systems perspective of the repair level problem.

Simulation models can be effective tools to evaluate various intergrated logistics management approaches because of their ability to treat complex interactions, time-dependent behavior, and system feedback mechanisms; however, they do not provide 'optimal' solutions but rather 'acceptable' solutions. Conversely, purely analytical approaches, while

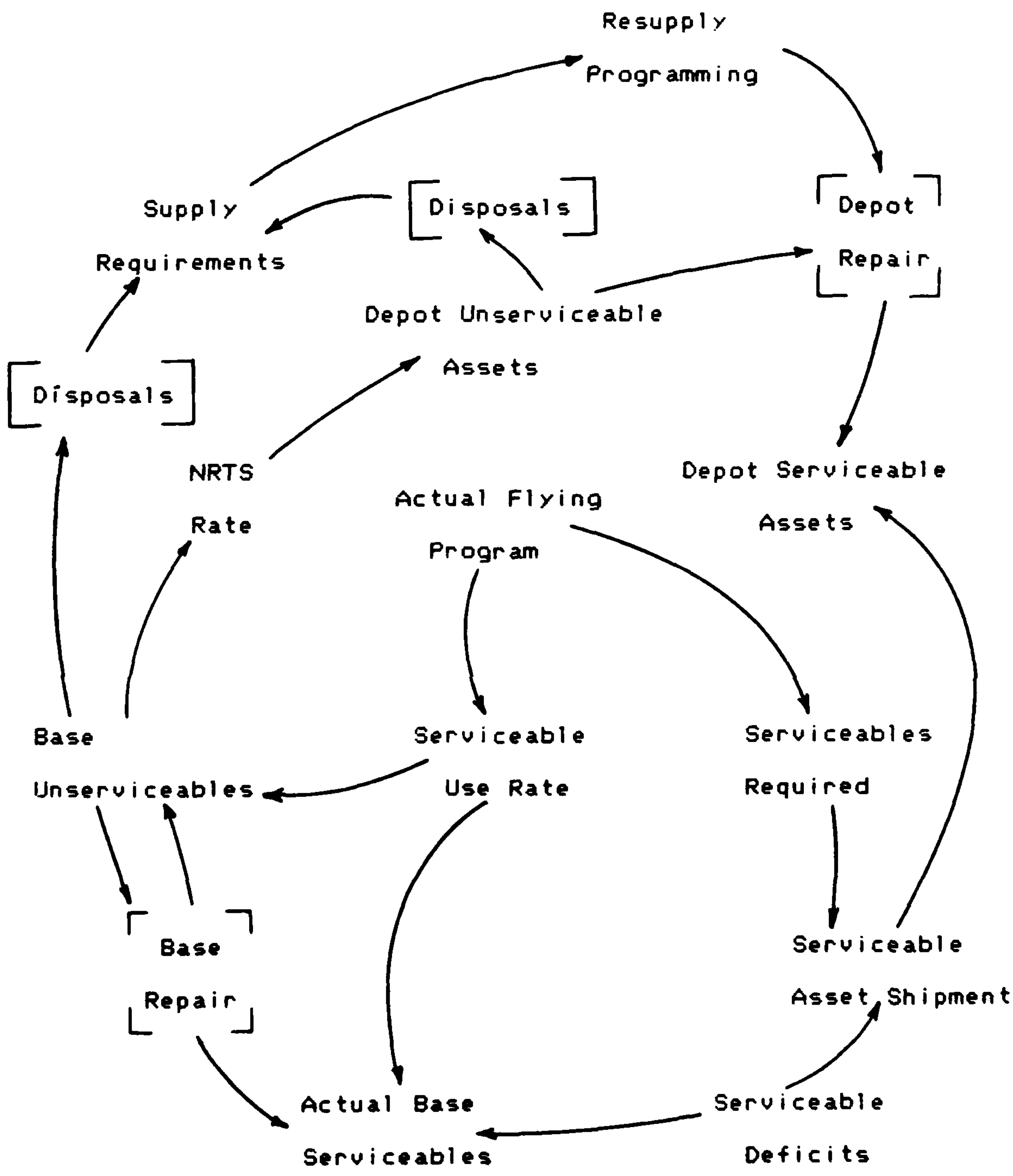


Figure 3. Reparable Assets Structure

proving accurate and appropriate for establishing stockage requirements during the initial provisioning process, are subject to criticism because of their limiting assumptions. The most damaging of these is that of applying steady-state approaches to dynamic systems. Practically speaking, once the system begins to operate structural shifts may occur requiring changes to the repair structure (22:391-392).

Aware of the inherent limitations experienced when applying analytical optimization techniques to dynamic systems such as the multi-item multiechelon repair system, the Air Force has initiated several efforts to assess, on a comparative basis, the life cycle costs (LCC) associated with selection of a repair level for a particular item within a system. Initial efforts were centered on single item repair analysis; however, as has been explained earlier in Chapter I, the inability to adequately address assignment of the fixed costs related to such items as support equipment severely damaged the creditability and validity of this technique.

During the early 1970s, the Air Force began to investigate the feasibility of using network theory as a basis for solving the repair level problem. Much of the motivation for this effort was generated by an article by J.M.W. Rhys' entitled, "A Selection Problem of Shared Fixed Costs and Network Flows" (20:3). Although Rhys does not specifically address the fact that the repair level decision problem could be structured as a special case of the shared fixed cost

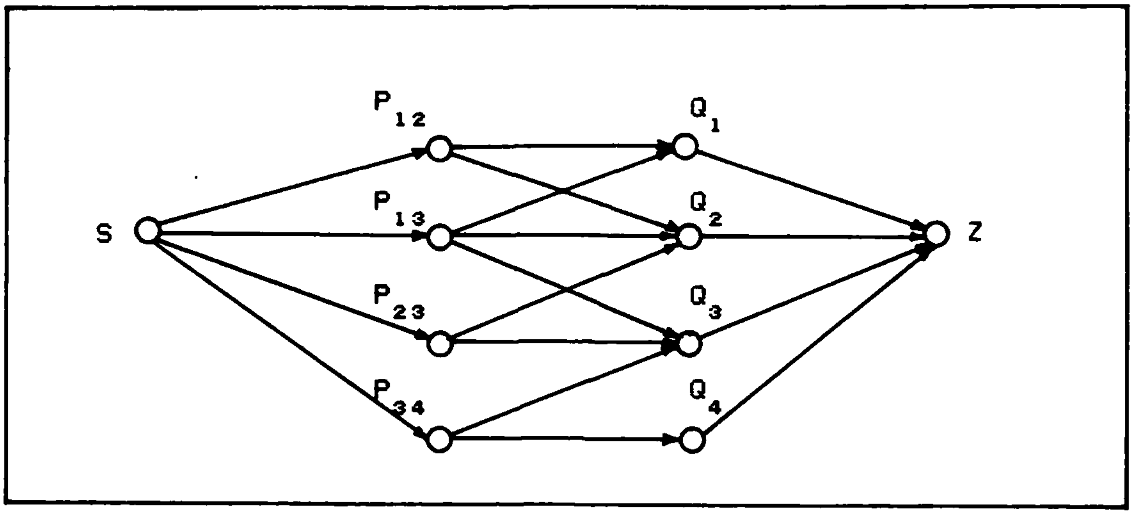


Figure 4. Shared-Fixed Cost Network

problem, he is generally credited with making the major transition from theory to application. The shared fixed cost problem can be described as the problem of selecting a set of activities from some larger set of activities, which require the allocation of fixed overhead costs that cannot be specifically related to one particular activity (21:201).

Using the duality principle of linear programming, Rhys hypothesized and proved that the shared fixed cost problem could be solved by creating a directed network as shown in Figure 4. By determining the maximum flow on this network the minimum cost decisions are simultaneously identifying the minimum cut set. The network has a single source, S and a single sink, Z. Sets of arcs [SP] and [QZ] are defined with each arc in SP corresponding to an activity (benefit), P, and each arc in QZ to a facility (cost), Q. Each arc in SP is allocated a capacity p, where p is the net gain associated with option P, and each arc in QZ is allocated a capacity c, with option P, and each arc in QZ is allocated a capacity c, where c is the net loss associated with option Q. (21:201).

The MITRE Corporation of Bedford, Mass. has successfully employed this technique on a variety of Air Force programs to identify optimum levels of repair for weapons systems items (20). Although MITRE's efforts were commendable, as late as 1978, a standardized software package implementing these procedures on an Air Force wide level was still missing. Being the office of primary responsibility (OPR) in the USAF for the development of analitical techniques such as this, the Concepts and Analysis Division, Air Force Acquisition Logistics Center (XRS/AFALC) initiated efforts to design, validate, and distribute just such a product. By June of 1980, a FORTRAN IV computer program known as the Network Repair Level Analysis (NRLA) program had been produced. In conjunction with this program, two documents, The NRLA Users Guide and The NRLA Programmers Guide, were published to assist in the understanding of the structure, limitations, and capabilities of the NRLA program. The Users Guide provides a general description of the model design, the programs operation in terms of input and execution requirements, an explanation of the LRU, SRU, and SE cost computations, as well as, a presentation of the network's formulation. The programmers guide is designed as an aid to understand the " programs structure, logic, input and output operations and the organization of data so that modification and/or corrections can be made " (2:1). One of the most useful portions of the programmers guide is Appendix A: the variables dictionary, which is an alphabetical listing of the FORTRAN variable names and arrays used throughout the NRLA program.

After reviewing the existing literature relating to the development and application of networking theory to repair level analysis in the Air Force, state-of-the-art solution algorithms were researched. These different techniques were investigated to determine the potential for and the appropriateness of their application within a specialized structure such as the RLA network. Initially the Max-flo Min-cut problem, which is the theoretical basis for the NRLA formulation, was studied. The various solution procedures which appeared to have potential are discussed in the following pages.

Max-flow Min-cut Problem

The max-flow min-cut theorem which was initially established in the late 50's and early 60's by Ford and Fulkerson is the basis for much of the success that has been experienced in formulating an allocation problem as a directed network. Given a network with a source node designated by 's' and a sink node designated by 't', the theorem is as follows: "For any network the maximal flow value from s to t is equal to the minimal cut capacity of all cuts separating s and t " (17:11). The following example should aid in understanding of this theorem. Consider the directed network shown in Figure 5. In this example the maximum flow on the network is 4 units, with 1 unit of flow on path s-1-

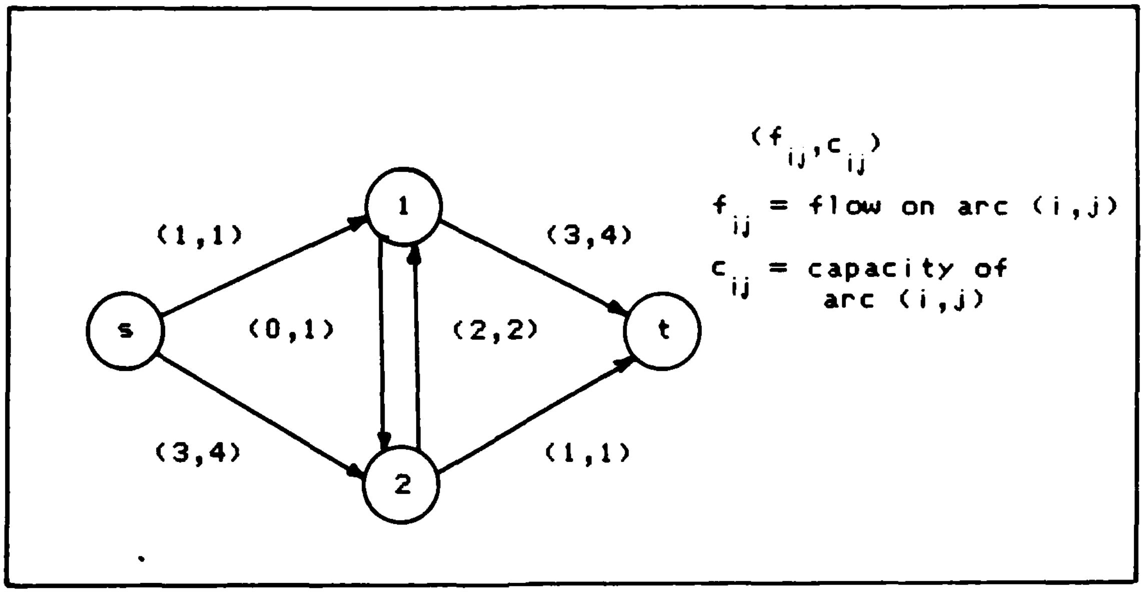


Figure 5. Example Max-flow Min-cut Network

s-2-1-t. The set of arcs (s,1), (2,1), and (2,t) has capacity 4 and is the minimum cut for the network. This set of arcs comprises the critical flow path in the network; to increase flow on the network the capacity of one of these arcs must be increased. Thus the capacity of the minimum cut is equal to the maximum flow.

Mathematical Formulation

The relationship between the max-flow problem and the min-cut problem is founded in the duality relationships of the two problems. The mathematical formulation of these problems will be presented here, for an in depth treatment of the duality relationships refer to one of the following works; Ford and Fulkerson (17:26-30), Bazaraa and Jarvis (5:473-477), or Jenson and Barnes (2:147-153).

<u>Max-flow Problem.</u> Consider a network with n nodes and m arcs, with each arc (i,j) having a lower bound of 0 and an upper bound (arc capacity) of c_{ij}. Let v represent the flow from node 1 (the source) to node n (the sink). Then the maximal flow problem can be stated as follows:

Maximize v

$$\text{Subject to} \quad \Sigma_{j=1}^{n} f_{ij} - \Sigma_{k=1}^{n} f_{ki} = \begin{cases} f \text{ if } i = 1 \\ 0 \text{ if } i \neq 1 \text{ or } n \\ -f \text{ if } i = n \end{cases}$$

$$f_{ij} \leq c_{ij} \qquad i,j = 1,2,\ldots,n$$

$$f_{ij} \geq 0 \qquad i,j = 1,2,\ldots,n$$

where the sums and inequalities are taken over all of the arcs in the network.

Before proceeding to the min-cut problem, the concept of a cut-set needs to be explicitly defined. Let N be the set of nodes contained in a network. Let X_1 be any subset of N containing node 1 (the source) but not node n (the sink). Similarly let X_n equal $N - X_1$, the subset of N containing node n but not node 1. Then the set of arcs connecting these two subsets is called the cut-set. For other than trivially small networks, to determine the number of possible cut-sets becomes a problem of a combinatorial nature. The min-cut problem seeks to identify the cut set with the minimum capacity.

<u>Min-cut Problem.</u> The dual to the max-flow problem is

the min-cut problem. Two additional variable sets are added in the dual, π_i which corresponds to the conservation equations and δ_{ij} which can be thought of as "identifying variables". They are identifying variables because in the optimal solution if $\delta_{ij} > 0$ then arc (i,j) becomes a member of the bottleneck set of arcs defining the cut-set. It is not necessary for the cut-set to be a unique set of arcs. The min-cut problem can be defined as follows:

$$\begin{aligned} \text{Minimize} \quad & \Sigma_{i=1}^{n} \Sigma_{j=1}^{n} c_{ij} \delta_{ij} \\ \text{subject to} \quad & \pi_n - \pi_1 = 1 \\ & \pi_i - \pi_j + \delta_{ij} \geq 0 \quad i,j = 1,2,\ldots,n \\ & \delta_{ij} \geq 0 \quad i,j = 1,2,\ldots,n \end{aligned}$$

Labeling Algorithms

The max-flow problem is usually solved using some type of labeling algorithm. There are a multitude of these algorithms available to solve this problem; however, many of these are the result of simply changing the decision rules for selecting the flow augmenting paths. The classical approach normally used to solve this problem, was developed by Ford and Fulkerson and uses a "Breadth First Search" algorithm as its basis. This approach has been fully developed and applied to the max-flow problem. Additionally, several refinements to this algorithm have been developed and will be discussed. Another approach to this problem bases its solution procedure on a "Depth First Search" algorithm as its basis. The conceptual basis for the depth first approach is presented in Horowitz and Sahni (19:268-269). However, the literature review did not reveal an application of this search algorithm to solving the max-flow problem; therefore an algorithm to accomplish this was developed as part of the thesis effort. A brief description of these algorithms will be presented with an in-depth description reserved for Chapter V.

Before proceeding to the labeling algorithms, a brief review of common terminology is in order. In both algorithms labels are assigned to the nodes of the network. Initially, all nodes except the source node are unlabeled. When a label is assigned to a node the label contains two pieces of information: the node the flow came from, and the amount of flow potentially available. Consider the label (1+,5), this label indicates that 5 units of flow is available from node 1. The label (2-,3) indicates that 3 units of flow which had previously been flowed to node 2 has the potential to be returned to another node for redirection. It should be noted that the labels only indicate the potential amount of flow which may pass between nodes. The actual amount of flow and the path it will take are not determined until a path to the sink node is found.

The node's label identifies the node as being in one of three states. State 1 is unlabeled, state 2 is labeled but unscanned and state 3 is labeled and scanned. The process of scanning occurs when a node has all of the arcs originating or terminating at it checked for available additional flow.

This is often referred to as checking for admissible arcs. Admissible arcs occur in two forms: forward arcs and reverse arcs. An admissible forward arc originates at the node being scanned, has a destination node which is unlabeled, and has current flow less than capacity. An admissible reverse arc terminates at the node being scanned, has a source node which is unlabeled, and has current flow greater than the lower bound on the arcs capacity.

Breadth First Algorithm. Initially, all nodes are unlabeled (state 1) except the source node which is labeled and unscanned (state 2). The source is then scanned and all admissible arcs are identified. Their associated nodes are labeled and their state changed to state 2. After each node is labeled, a check is made to determine if the sink node has been labeled. If it has, the algorithm goes to a flow augmenting procedure; if not, it continues. When the source has been scanned, a labeled but unscanned node is selected (usually in numerically ascending order) and is scanned. This continues until the sink node is labeled or there are no available nodes in state 2. If there are no nodes in state 2, then the maximal flow has been identified and the algorithm stops. When the sink node is labeled, the flow augmenting algorithm comes into play. It identifies the amount of flow which could be flowed to the sink from its label. It then uses the labels to trace the path back through the network to the source. Each arc on this path has its flow adjusted by the amount which actually reached the sink node. Once this flow augmentation is complete all labels are erased and the algorithm restarted.

Since arc capacities are restricted to being only integer values, termination of this algorithm will occur in a finite number of iterations. Ford and Fulkerson (17:121) provide an example that shows that the algorithm may not terminate or terminates with a sub-optimal flow when irrational arc capacities are allowed. However, a problem can still arise when using integral capacities. This involves the situation where the total number of flow augmentations is equal to the maximum flow on the network. Edmonds and Karp (16:250) present an example where the flow augmentation at each iteration is only one unit, this leads to an upper bound being created for the number of augmentations equal to the max-flow of the network.

Two refinements of the algorithm are presented by Edmonds and Karp (16:251-253) which can reduce the upper bound on augmentations required. The first refinement involves decision rules for selecting the shortest paths for flow augmentations. They show that if each flow augmentation is done along a path with the fewest number of arcs then a max-flow will be identified in at most $1/4(n^3-n)$ augmentations. This result is derived based on the length of the paths to a bottleneck arc from the source and the sink, and the max number of occurances of bottleneck arcs in a network of n nodes. A further refinement to this process, includes using a rule where the weights of the paths are calculated based on their lengths and these weightings are then used to aid in the selection of a path. Another rule involves selecting the path with the fewest number of reverse arcs.

A second refinement presented uses a decision rule which selects the path with the maximum possible number of flow augmentations (16:253-255). This refinement provides that the maximum number of augmentations necessary to attain a maximal flow will be at most $1+\mathrm{Log}_{M/(M-1)} f^{*}(t,s)$, where $f^{*}(t,s)$ denotes the value of maximal flow. Both of these refinements are intuitively appealing in that the selection of a shortest path or the max-flow per augmentation seem to be good rules of thumb for obtaining max-flow in the fewest number of iterations. Additional considerations for speeding up the process could involve: not erasing the existing set of labels after each iteration, or simply saving the location of the last node with excess potential on the flow augmenting path. This could save a great deal of time which is expended doing redundant labeling and should significantly improve computational times.

<u>Depth First Algorithm.</u> As in the "Breadth First Algorithm" the initial states of the nodes in the "Depth First Algorithm" are source node in state 2 and all other nodes in state 1. The search for a flow augmenting path initiates at the source by identifying an admissible forward arc. The node associated with this arc is labeled and its state is changed to state 2. A search is then made from this node for an admissible arc and the process is repeated until

the sink is labeled or no admissible arcs are available to augmenting program is called and flow is augmented along the path. As the flow is augmented on each arc a check is made to determine if there is any potential flow remaining at any of the nodes on the path. If there is no flow potential at a node, that node's label is erased and it is returned to state 1. When this augmentation is complete, the algorithm then starts with the last labeled node on the path and proceeds to find another path to the sink. If the sink cannot be reached from the last node on the path; and there are no admissible arcs at this node, then the algorithm backs up to the previous node on the path and checks for admissible arcs. This process continues until the sink is labeled or until the source node is revisited, at which point a new admissible arc is chosen from the source and the algorithm starts again. When there are no further admissible arcs at the source and the last path has been retraced back to the source, the maximal flow has been found and the algorithm halts.

To ensure that the maximal flow is obtained and to identify the unique set of labels associated with this flow, it is necessary to make three passes through the network with this algorithm. During the first pass only forward arcs are considered. The result of initially restricting the flow along just these arcs is that the majority of the flow reaches the sink node via the shortest available paths. Additionally, it creates the potential for redirecting flow on a reverse path on the second pass. The labels are all erased at the end of the first and second passes so that all possible paths can be explored during the subsequent passes. The second pass is used to identify all additional paths through the network on which flow can be augmented. These paths may contain reverse arcs. The third pass is used to identify the unique set of labels associated with the minimum cut set.

Minimum Cut Algorithm. As explained earlier, the optimum set of repair level decisions can be determined by finding the minimum cut set of the RLA network. Because of this relationship, an algorithm which could directly solve the min-cut problem would be acceptable for use with RLA networks. A search of the literature indicated that the most promising algorithm of this type was developed by Dessouky and Phillips (15) and is called the "Cut Search Algorithm". In this algorithm the min-cut set is located directly by a cut seeking procedure.

The min-cut set, which is designated by $K^*(s,t)$, is identified by a two stage process which divides the set of network nodes, designated by N, into three groups T, W, and S. The first stage of the process starts by assigning the sink node to set T. Set T is then expanded toward the source by computing the value of its cut set defined by $K(\bar{T},T)$ where $N = \bar{T} + T$. At each iteration a calculation is performed for every node which is a member of $\bar{T}$ and has an arc connecting it to T to determine if it can be added to T. This stage terminates when $\bar{T}$ contains only the source node or if no nodes in $\bar{T}$ can be added to T. If $\bar{T}$ contains only the source node then $K(\bar{T},T) = K^*(s,t)$, and the minimum cut set has been identified. If T cannot be expanded to include all nodes except the source then the second stage is initiated. In this stage the sets W and S initially contain only the source node. Set W is expanded toward set T until $\bar{W} = T$ at which point the min-cut set is contained in S. S is contained in W; however, it is expanded only by adding nodes of W, which will reduce the value of the cut-set as defined by $(S,\bar{S})$. Therefore at termination $K^*(s,t) = K(S,\bar{S})$.

Computational results comparing this algorithm with the Out-of-Kilter Algorithm (OKA) developed by Fulkerson and Danzig are presented in Dessouky and Phillips' article (15:403). These results indicate that the Cut Search algorithm is more efficient than the OKA. As an example; for a network consisting of 16 nodes and 240 arcs the OKA took 5.33 seconds of cpu time while the cut search algorithm took 1.58 seconds (15:403).

Although the Cut Search Algorithm appears to be very efficient, investigation of its applicability to the RLA problem was not addressed during this research effort. This algorithm was reserved for investigation in the event that the depth first labeling algorithm proved less efficient than anticipated and to allow for the investigation of the applicability of generalized network primal algorithms described in the next section.

Generalized Network Primal Algorithms. In recent years significant advances have been made in the development and implementation of a class of algorithms which can solve large scale capacitated transportation and transshipment problems. These algorithms, often referred to as primal network codes, can also be applied to the solution of the max-flow problem. Because of this characteristic and the potential for applying NRLA to large size RLA systems such as the B1-Bomber and MX-Missile programs, investigation of one of these primal network codes seemed appropriate. The code chosen for use in this context was a commercially available code called GNET. This code was selected for two reasons; (1) it was currently available on the computer systems accessable to AFIT and (2) it was the subject of an excellent article in Management Science by Bradley, Brown and Graves (7) which covered in detail its design, implementation and use. This type of in-depth coverage of a commercially marketed algorithm is usually very difficult to find or is non-existent.

Due to the complexity of the GNET code, and the fact that it was applied as a subroutine in NRLA to solve the max-flow problem only a brief description of the GNET code will be given. For an in-depth explaination of this particular application of primal networking one should refer to the above referenced article. Basically, the GNET code solves the general linear programming (LP) problems (transportation, transshipment, and max-flow) by specializing these LP problems into a primal network model. The basis for solving this network is the bounded variable revised simplex method. The uniqueness of this approach however, lies in the fact that the network which is developed by the algorithm takes advantage of the sparsity of the arc node matrix and uses upper triangularized basis representations to quickly solve the network.

Results published by Glover and Klingman indicate that when this code is used for finding solutions to transportation and transshipment problems, that it is 30% to 40% faster than the OKA code in the solution of transportation and transshipment problems. Bradley, Brown and Graves indicate that it is currently believed that primal implementations are faster and require less storage than OKA or other algorithms used to solve these type problems (7:3).

III. Methodology

Introduction

Initially, a study was made of the logic and rationale for each part of the NRLA program. The NRLA program, originally developed on the AFLC Honeywell 635 computer using the CREATE time sharing system is "composed of a main routine, a block data subroutine, plus twelve additional subroutines" (2:1). The logic diagram in Figure 6 displays the interrelationships of the various subroutines to the main program. Appendix B provides a brief explaination of their respective functions. After several discussions with both the developers and users of the model (6,9), it was determined that wherever possible a structured programming approach using FORTRAN subroutines would be used to incorporate new or enhanced features into the model. By using this type of approach, there would be minimal impact on current users of the model. Additionally, it was recognized that a strategy of this type would probably facilitate the verification and validation phases of the design process. With this general concept in mind, the identification of the necessary tasks related to accomplishing the thesis objectives was initiated. In conjunction with this effort, a flow diagram was developed to order and prioritize these tasks. This was done to better visualize how they the total process. This conceptual process resulted in the flow diagram shown in Figure 7. To successfully achieve the thesis objectives of:

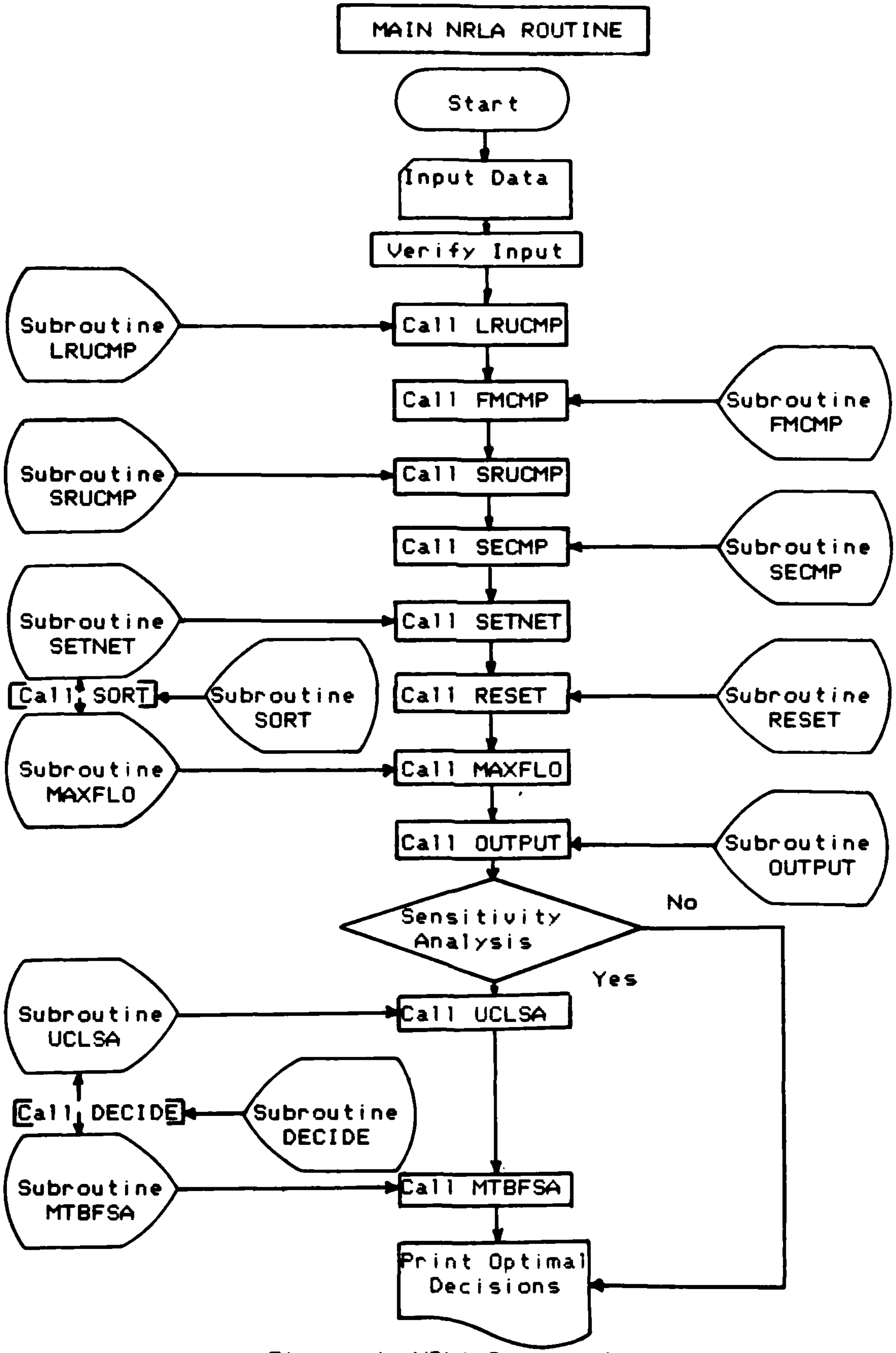


Figure 6. NRLA Program Logic

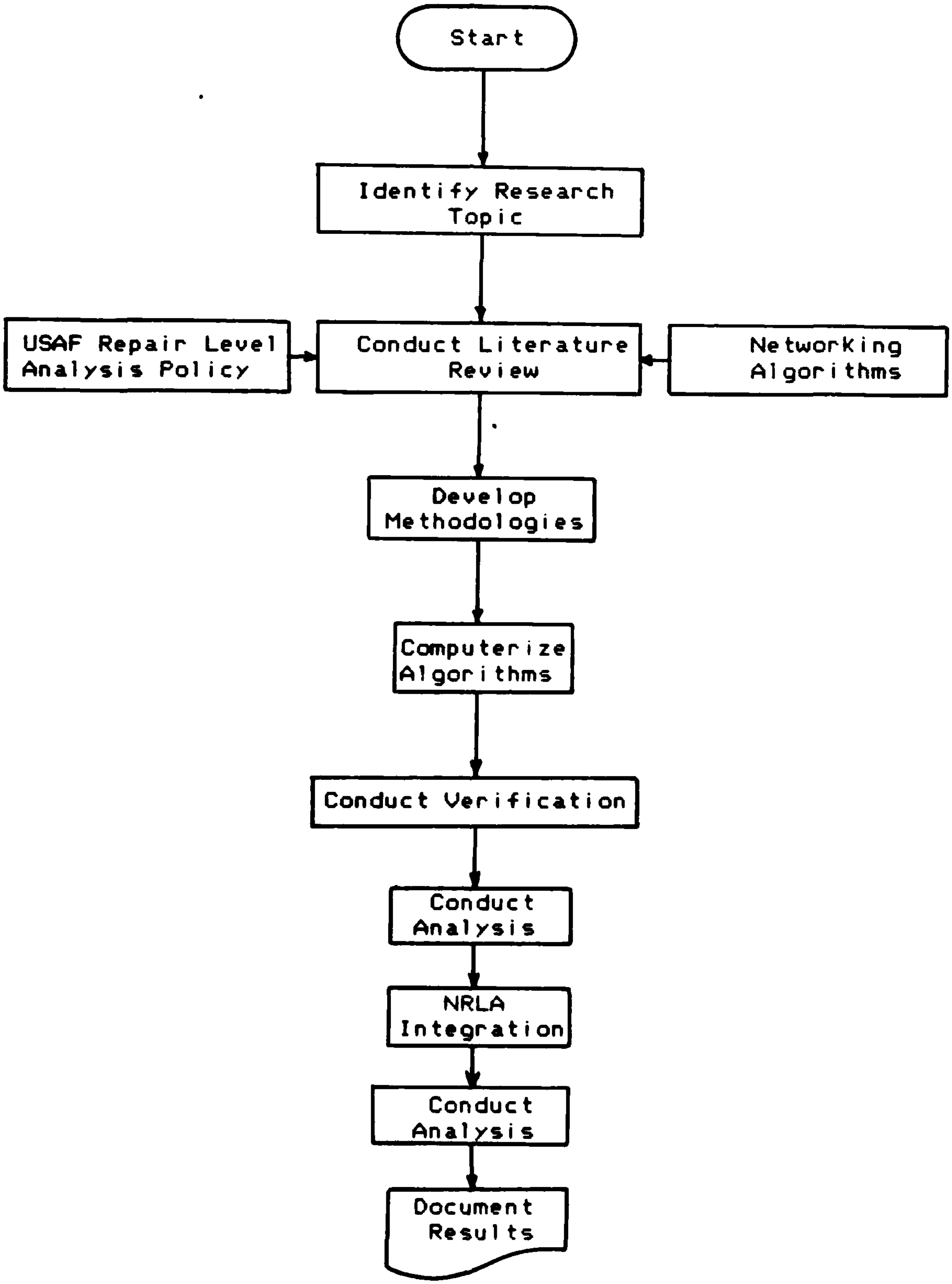


Figure 7. Thesis Task Organization

reducing the NRLA program's execution time, and integrating the CIRF repair level option, four new subroutines were designed and evaluated for possible integration into the existing NRLA structure. These are called MARLA, MSETNT, NMXFLO, and CRFSET and are indicated by the dashed lines in Figure 8.

The MARLA subroutine employs the concept of marginal value analysis in an attempt to determine if optimal decisions can be made for individual items. As can be seen from Figure 8, the MARLA subroutine functions as a preprocessor or filter to the MSETNT subroutine. In effect, when MARLA makes an optimal decision for an LRU or SRU failure mode, it has reduced the total number of items which need to be further analyzed. It was believed that gains in time efficiency would be achievable from this enhancement; the network could be built faster in MSETNT, and the problem to be solved in MAXFLO would be of smaller magnitude. This assumes that the execution time requirements for the MARLA, MAXFLO, and MSETNT subroutines would be less than what is currently required for the SETNET and MAXFLO subroutines in NRLA.

The second area which exhibited a potential opportunity for reducing NRLA's execution time was in the MAXFLO subroutine. MAXFLO's purpose is to determine the minimum cost set of optimal repair level decisions. The solution technique currently employed is a direct application of the two stage labeling procedure developed by Ford & Fulkerson (17:22). Implementation of a new labeling procedure appeared likely to

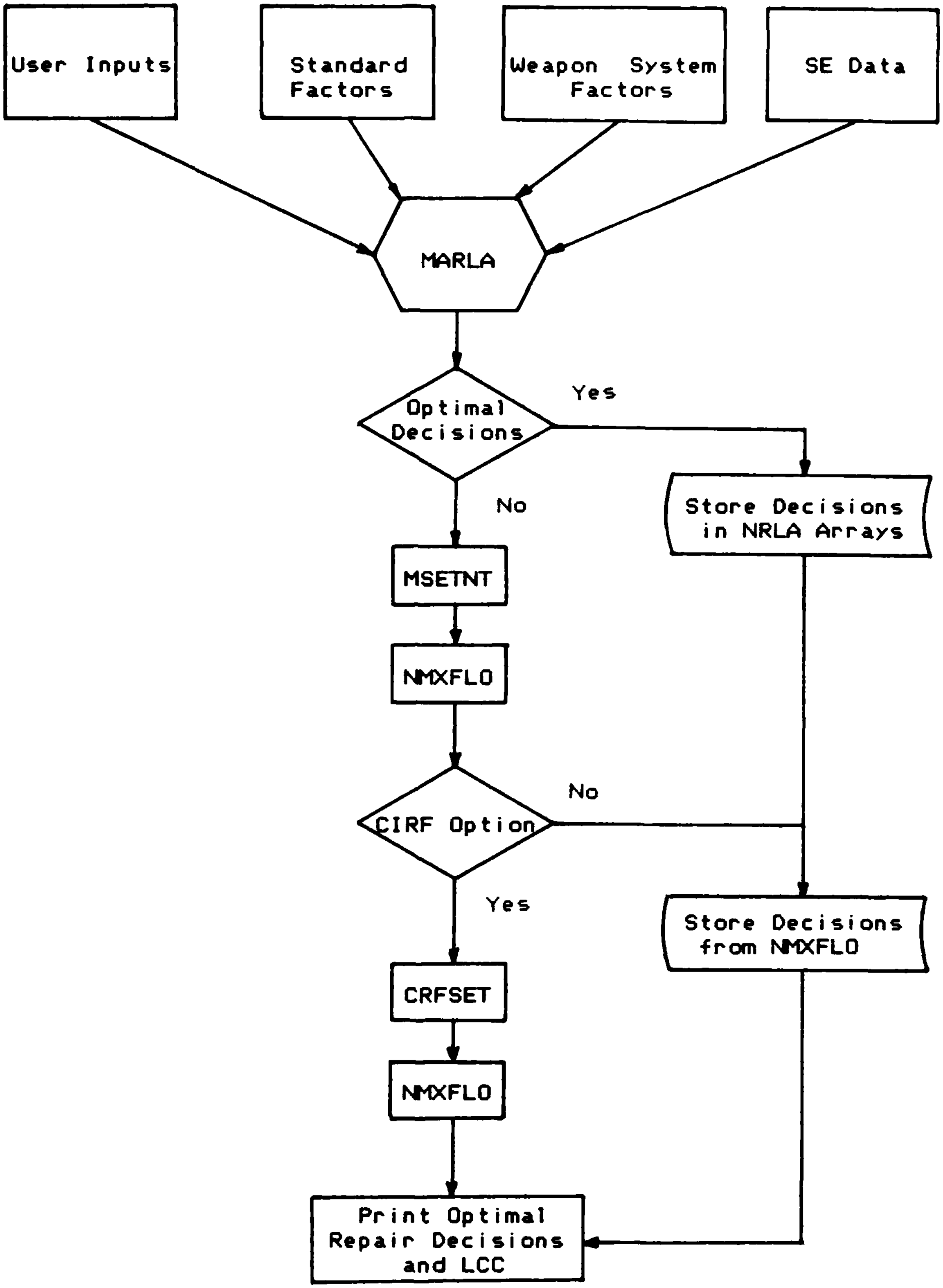


Figure 8. NRLA Thesis Methodology

yield significant time savings in this area of the program. The alternative labeling technique which was developed is constructed in NMXFLO and, as will be explained later, can be used as an alternative to, or in conjunction with, the existing MAXFLO subroutine.

The final objective of incorporating a CIRF maintenance concept as a fourth repair alternative was proposed to be accomplished by using a third subroutine called CRFSET. Subroutine CRFSET was used with the existing SETNET subroutine to determine if the CIRF repair option could be the optimal decision for a particular item. It compares the repair level recommendation generated by SETNET, (depot, base, or scrap) with the CIRF option by building a new unique network (See Figure 8).

Although integration of each of these subroutines would necessitate modifications to other portions of the NRLA program, variable and array integrity was to be maintained to the maximum extent possible. Additionally, the individual subroutines were developed with the objective of being compatible with existing program data structures, as well as, with linkages between data elements (i.e. the use of pointers). Separately, these subroutines accomplish specific subobjectives which contribute to the successful achievement of the previously outlined thesis objectives. The following three chapters will discuss in detail the concept development, computerization, verification, validation, and analysis phases of each of these three major enhancements.

IV. Marginal Analysis Enhancement

Concept Development. The application of marginal cost analysis techniques to solve the repair level decision problem is conceptually based upon the idea that the analyst can use existing cost information that is readily available to quickly identify the optimal level of repair for a specific item. AFSC/AFLC Pamphlet 800-4, Repair Level Analysis Procedures, identifies marginal analysis as a simple and efficient way to synthesize the repair level problem into a more workable format (13:32-33). An appealing feature of using marginal analysis for repair level decision making is its ability to make decisions without prorating the cost of the support equipment which would be necessary to repair an item. This precludes the possibility of making inconsistent or suboptimal decisions. It is for these reasons that development of a marginal analysis approach was selected as a possible solution to solving the problem associated with NRLA's computer run time.

To repair or replace an item or component of an Air Force system requires the expenditure of materiel and personnel resources, as well as the utilization of specialized and sophisticated diagnostic test equipment. The costs which are generated from this process can be broadly categorized into two types of costs: (1) Pipeline costs (P) - which are defined as those costs incurred directly to repair the item, and (2) Support Equipment costs (SE) - which are defined as

Table II

Repair Level Costs

I. Pipeline Costs

A. Initial Spares
B. Packing and Shipping
C. Replenishment Spares
D. Repair Labor
E. Item Management
F. Technical Management
G. Training of Maintenance Personnel

II. Support Equipment Costs

A. Acquisition
B. Operation and Maintenance
C. Facilities

III. Scrap Costs

A. Initial Spares
B. Replenishment Spares

those costs related to SE use that cannot be assigned to a specific item. Table II gives an itemized listing of each of these categories of cost. Once these costs have been determined for each possible level of repair, a marginal analysis can be made of the SE resources at each repair location. In the context of comparing the four repair options of Depot, Base, CIRF, or Scrap, if the scrap cost of the item is known to be less than the pipeline costs of the item at the other locations, then the scrap decision will always be the optimal decision for that item. In this situation, the scrap decision is said to dominate the other three alternatives (13:32). Equations (1) through (3) represent the mathematical interpretation of the marginal analysis comparisons.

$$\text{If } P_S < P_D \quad \text{then} \quad P_S < P_D + SE_D \tag{1}$$

$$\text{If } P_S < P_C \quad \text{then} \quad P_S < P_C + SE_C \tag{2}$$

$$\text{If } P_S < P_B \quad \text{then} \quad P_S < P_B + SE_B \tag{3}$$

Where:

P_S = pipeline costs to scrap the item.

P_D = pipeline costs to depot repair the item.

P_C = pipeline costs to CIRF repair the item.

P_B = pipeline costs to base repair the item.

SE_D = cost of the depot support equipment.

SE_C = cost of the CIRF support equipment.

SE_B = cost of the base support equipment.

In order to determine the marginal value of the SE resources for each level, the pipeline costs associated with repair of the item are subtracted from the pipeline costs to scrap the item:

$$MV_{SD} = P_S - P_D \tag{4}$$

$$MV_{SC} = P_S - P_C \tag{5}$$

$$MV_{SB} = P_S - P_B \tag{6}$$

From the above computations, it can be seen that in this initial example that the marginal value of the SE at every level was negative. This would mean that there would not be any cost savings to be realized from buying the necessary SE to repair the item. In fact, before the purchase of SE resources could be economically justified, either the pipeline scrap cost would have to increase or the pipeline costs of one of the repair options would have to decrease until one of the SE marginal values became zero.

Unfortunately, many situations do not lend themselves to such a straight forward type of analysis. A much more comprehensive set of decision criteria must be developed when dealing with SE that possess a positive marginal value. Based on the scenario of having three possible repair locations, as well as, the option to discard the item, a comparison must now be made not only between the pipeline costs of the item; but also between the total costs to repair the item for each of the alternative levels. Table III displays in a systematic fashion the necessary cost comparisons which must be made to ascertain whether or not a repair option can be eliminated. If any of the statements are true for a given repair level, that particular repair option is eliminated. To interpet the meaning of these relationships with respect to the marginal value of the SE, a joint marginal value is calculated as follows:

$$MV_D = \text{Min } [(P_C - P_D), (P_B - P_D), (P_S - P_D)] \quad (7)$$

$$MV_C = \text{Min } [(P_D - P_C), (P_B - P_C), (P_S - P_C)] \quad (8)$$

$$MV_B = \text{Min } [(P_D - P_B), (P_C - P_B), (P_S - P_B)] \quad (9)$$

The joint marginal value is always taken as the minimum of the three values because it identifies the maximum amount of funds that should be allocated for the SE at that repair location. As soon as the cost of the SE exceeds this amount,

TABLE III

Marginal Analysis Decision Criteria

DEPOT

If ($P_S < P_D$) then D′

If ($TC_C < P_D$) then D′

If ($TC_B < P_D$) then D′

CIRF

If ($P_S < P_C$) then C′

If ($TC_D < P_C$) then C′

If ($TC_B < P_C$) then C′

BASE

If ($P_S < P_B$) then B′

If ($TC_D < P_B$) then B′

If ($TC_C < P_B$) then B′

Where : D′ = eliminate the depot repair option.

C′ = eliminate the CIRF repair option.

B′ = eliminate the base repair option.

$TC_D = (P_D + SE_D)$ total repair costs at depot.

$TC_C = (P_C + SE_C)$ total repair costs at CIRF.

$TC_B = (P_B + SE_B)$ total repair costs at base.

it would no longer be cost effective to purchase the support equipment to repair the item (13:71). A deductive type of process is used to isolate the optimal decision. Only if three of the four choices can be eliminated does the marginal analysis approach identify the optimal decision. However, if marginal analysis were to be used as a separate analysis program the utility of its approach shhould not be based solely on its ability to identify optimal decisions. But rather a more accurate measure would be the total number of repair options it could exclude from the decision making process.

Computerization/Integration. To implement the concept of marginal repair level analysis a FORTRAN IV program called MARLA was developed. Appendix D contains a computerized listing of the MARLA subroutine. This subroutine is designed to be called by the MAIN subroutine subsequent to the computation of the LRU/SRU pipeline repair costs which are performed in the LRUCMP, FMCMP, and SRUCMP subroutines, but prior to the computation of the SE requirements which are made in subroutine SECMP. The reason that the MARLA subroutine should be called in this manner is two fold: first, to accurately perform the marginal analysis MARLA needs the pipeline costs for each of the items, and second, any SE that is purchased as a result of a MARLA optimal decision should be considered when determining the additional SE requirements for the remaining items.

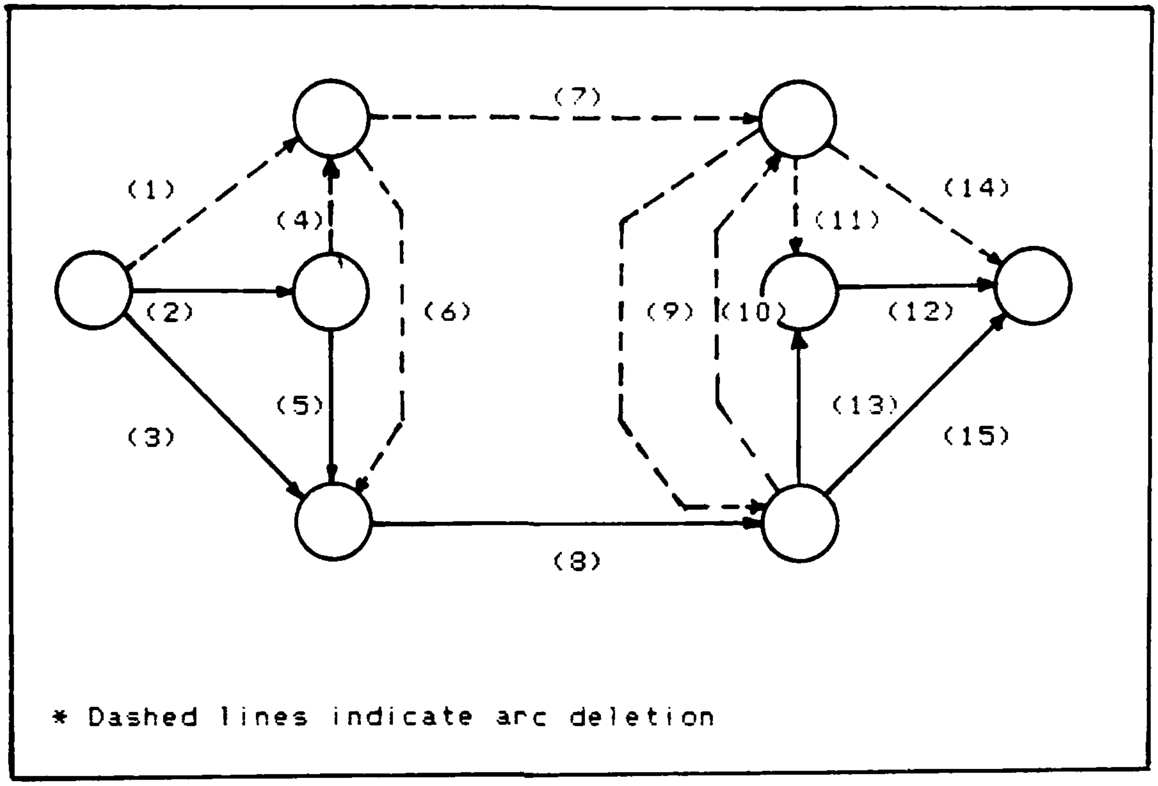


Figure 9. Example of Proposed Arc Elimination Process

The most difficult interface that had to be designed for integration of the MARLA subroutine was the modification of subroutine SETNET to accept the MARLA decisions. When optimal decisions were made by MARLA, it eliminated the requirement to construct the arcs which would be necessary in MAXFLO to analyze the item. The example in Figure 9 indicates the arcs that could be erased from a network if an optimal decision is made; in this case if an optimal decision had been made for LRU #1, eight out of fifteen of the network arcs (#1,#4,#6,#7,#9,#10,#11,#14) are no longer needed. Using this idea, modifications were made to subroutine SETNET to give it the capability to reconfigure a new network based strictly on just those items still requiring a repair level decision. This new subroutine was called MSETNT. By implementing this change, greater efficiencies were expected to be realized during the programs operation. The code which was developed to accomplish this task is contained in Appendix E.

Because item unique information such as unit cost and SE utilization hour requirements are stored in separate arrays for LRU and SRU failure modes, MARLA was divided into two major sections of code; one to analyze LRU items and another to handle SRU items. Proper control of the NRLA exclusion arrays throughout both of these portions of the program was determined to be extremely important when using MARLA as a preprocesser to NRLA. There were three primary reasons for this. First, if a repair level had been excluded by the user, it is not necessary to perform a marginal analysis for that particular combination of item and repair level. Second, when the actual network is constructed in subroutine SETNET, any repair levels that have been excluded have their respective arc costs set to a very large cost called JUMBO. This prevents them from ever entering the cut set of optimal decisions. However, the MARLA decision criteria were not sufficient to exclude a repair option from further consideration by NRLA. Consequently, the FORTRAN code which was written to accomplish the marginal analysis was reequired to embody this situation. The third effect MARLA's integration had on the exclusion arrays occured in subroutine OUTPUT.

Here the exclusion arrays are checked again to see if the user has eliminated any of the repair options. For those items where repair levels have been omitted, the program will print the word 'XCLD'. Two programming changes were required in subroutine OUTPUT to ensure that this operation was performed for only user exclusions and not for exclusions generated by MARLA.

Another important aspect of subroutine MARLA is the identification and acquisition of the correct quantities of SE resources necessary to repair the item. Although when performing the marginal analysis the potential SE costs are calculated for each item, the actual procurement of these resources should not be made unless an optimal decision has been made for that item. For this reason the logic of the MARLA subroutine does not adjust the SE arrays for cost and hours until the optimal decision arrays, OPDECL and OPDECS, are screened to determine if in fact a valid requirement does exists for that particular piece of support equipment. MARLA is structured to buy the minimum number of types of SE needed for each repair level based on the total system requirement. This was accomplished by using the logic described in Figure 10. The pivotal factor in this procedure was the accurate determination of the SE hours currently available for each type of SE at each repair level. This precluded the purchase of more SE than was actually needed. The array, HRAVSE, was created and integrated into the common MARLA block statement to store the available hours for each piece of SE and to

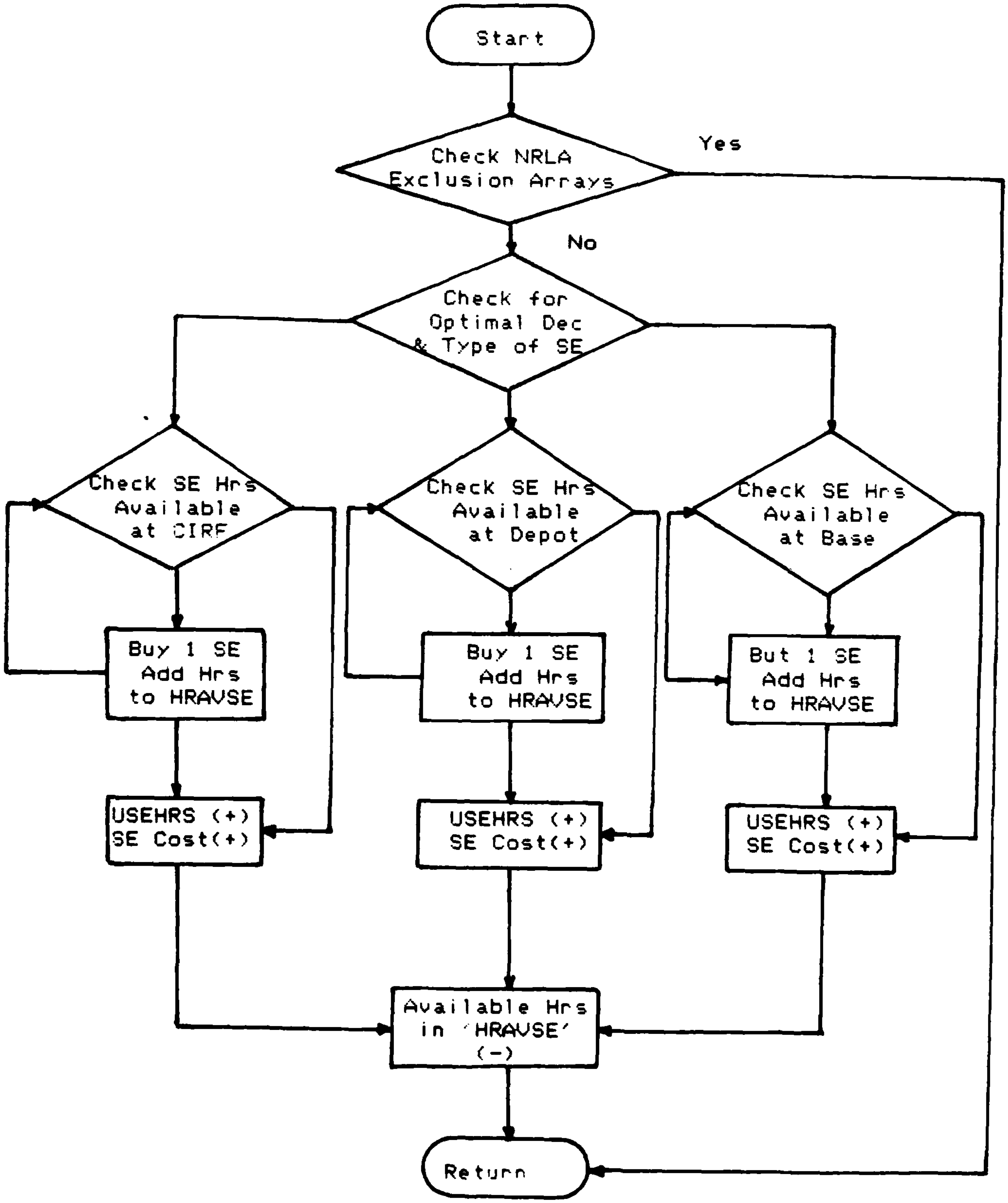


Figure 10. MARLA SE Requirements

allow for these hours to be transfered to the subroutine, SECMP. This ensured that any SE hours still available could be applied against the utilization hours required for the remaining items to be analyzed.

Verification/Validation. To verify the operational accuracy of the MARLA enhancement, two undertakings were necessary. Not only was it essential to confirm the logic of the MARLA subroutine, but it was also critical to separately authenticate the major changes that had to be made to the SETNET subroutine. It was decided that these tests should be conducted independently to provide the maximum assurance that both portions of the program were performing correctly.

Initial testing of the MARLA subroutine revealed significant program logic flaws. By modifying the subroutine so that it could be run as a separate program errors were quickly identified and corrected. This would not have been possible if MARLA had been immediately integrated into the NRLA program. The sample data set which was used for this testing included a total of 18 LRU/SRU failure modes with three different types of SE at each of the repair locations. Prior to running the program, the correct repair decisions were manually calculated using the decision criteria in Table III on page 46. The results from the MARLA program were then compared to these recommendations. This iterative process continued until the MARLA decisions matched identically with the true decisions. The results obtained from this test indicated that the MARLA subroutine was functionally correctly both computationally and logically.

A different design was developed to verify the accuracy of the SETNET modifications. A predefined network was utilized to determine if MSETNT would eliminate the the appropriate arcs for a given LRU/SRU optimal decision. By presetting the OPDECL and OPDECS arrays with a particular optimal decision, the arcs to be eliminated from the network could easily be identified. A trace of the network, consisting of each arc's source node and destination node, was output to determine if the correct arcs had been removed. Initially, the program eliminated only a portion of the arcs it should have. However, by continuing to use the trace from the network, all of the logic and syntax coding errors were satisfactorily resolved.

During this process, an important point was discovered. If the optimal decision for an LRU is 'base', the arcs representing the transportation costs for replacement SRUs (#6,#9,#10 in Figure 8, pg.48) could also be eliminated. However, since the costs associated with these arcs are related to the repair of the SRU, it was realized that they should be added to the SRU depot and scrap repair arcs (#3 and #8, Figure 8) so that the SRU costs would not be underestimated.

Initial validation of both the MARLA and MSETNT subroutines was accomplished by consulting with the original developers of the NRLA program (6). However, complete validation of the MARLA enhancement will only be determined after

NRLA users throughout the Air Force and defense contracting community have had the opportunity to exercise this option and provide feedback as to its performance and accuracy.

Analysis/Results. The feasibility of using the MARLA subroutine as a preprocesser to the NRLA program was analyzed in terms of its effect upon total processing time and additional storage requirements. A data set which had previously been analyzed by the original NRLA program was used with the enhanced program. In this way, a standard of comparison could be established to measure the effect on program time and storage requirements. Additionally, the repair level recommendations produced could be compared for consistency.

When used as external preprocessor, the MARLA program produced results identical to the NRLA program. However, when integrated as a subroutine in the main program, the repair recommendations generated did not agree. It was observed that although the pipeline logistical costs for each repair level matched exactly for both programs, the SE cost estimates differed significantly. It was determined by cross checking the outputs from both programs that this was problably the reason for the disparity in the model's recommendations.

Two other significant results were observed from comparing the output of the two programs. These related to the processing time and the number of arcs constructed in each of the networks. These program parameters are directly proportional to each other; the greater the number of arcs in the network, the longer it take the program to execute. For the sample data set, which consisted of 23 LRU failure modes, 10 SRUs, and 12 pieces of SE, the original NRLA network consisted of 361 arcs and required .0149 hours to execute. For the enhanced program the output indicated that the network contained only 12 arcs and processed in .0034 hours. Although these results showed a substantial improvement over the original network, a high level of confidence could not be placed in their validity because of the programs inability to produce decisions consistent with the original NRLA program.

Storage requirements for the new program were also analyzed. To incorporate the MARLA subroutine nine new arrays were required. Seven of these were needed to perform the marginal analysis while the two remaining ones, HRAVSE and TSECST, were used to collect the SE cost and available hours, respectively. The number of arrays was a function of the number of repair locations. The size of the arrays used in the marginal analysis was determined by the total number of failure modes and SRUs to be analyzed. Appendix A contains a listing of each of these arrays and their function in the program.

Based on the limited test results which were achieved with MARLA during this study, further efforts are necessary in order to fully assess its capability as a preprocessor to the NRLA networking process. A better evaluation could then be made of its utility in terms of reducing the overall program processing time versus the additional storage requirements to achieve this.

V. Max-flow Min-cut Labeling Enhancement

NRLA Network Formulation

Network Structure. Understanding the operation of the NRLA program requires an in-depth knowledge of its network structure and how this structure models the cost factors associated with the repair level decisions. The ten decision cost factors and the eleven logistics cost factors which comprise them are shown in Table I on page 8. As can be seen in this table, the decision cost factors are simply the summation of the logistic costs associated with a LRU's or SRU's repair level decision. This consolidation of costs allows their use as capacities for the arcs of the repair level network. The network formulation was examined briefly in Chapter I; however, the details of how the network is built and how the various cut-sets (repair level decisions) are determined were not covered at that time. The following section covers these topics in detail.

The NRLA program builds the network structure by first assigning numbers to the nodes in a specific order and then laying in the arcs associated with these nodes. As shown in Figure 1 on page 7, the nodes are assigned numbers in the following order source node, depot support equipment, depot LRU failure modes, depot SRUs, base LRU failure modes, base SRUs, base support equipment, and finally the sink node. This careful ordering of the nodes allows the program to develop forward and backward pointers to the vector arrays containing the LRU, SRU, and SE relationships.

Next the arcs are added to the network. This is also done in an ordered manner starting with the source node and working through the nodes one at a time. For simplicity the LRU failure mode nodes will be refered to as LRU nodes during this discussion. The arcs from the source to the depot SE (depot SE cost) are added first, followed by the arcs to the depot LRUs (depot LRU repair costs) and the SRUs (depot SRU repair costs). The next arcs added are the depot SE to LRU and SRU JUMBO (large capacity) arcs. These arcs shown as the thicker arcs in the figure represent the relationships between the individual depot SE and the set of LRU/SRUs which it repairs. Because these arcs have JUMBO capacity they cannot be capacitated (flow equals capacity); therefore, they allow the total set of LRU/SRUs to share the burden of the SE cost.

The arcs originating at the depot LRUs are added next. First the depot LRU to depot SRU (DEC1) arc is added. Then the depot LRU to base LRU arc (scrap cost). The DEC1 arcs represent a portion of the additional costs associated with the decisions to base repair the LRU and depot repair the SRU. The rest of these costs are contained on the DEC2 arcs. The costs are split to allow for representation of the two way trip associated with SRU depot repair verses the one way trip if the SRU is scrapped when the LRU is base repaired.

The SRU scrap cost arcs are added next. They originate at the depot SRU and terminate at the base SRU. Moving on to the base LRU nodes the following arcs are added. First an arc from the base LRU to its associated base SRU (DEC2 costs) and then an arc from the base LRU to the sink node (base LRU repair costs). Following these the arcs emanating from the base SRUs are added. The first arc is a JUMBO arc from the base SRU to its associated base LRU. This arc ensures that the decisions to base repair the SRU and scrap or depot repair the LRU cannot be made. Next an arc from the base SRU to the sink node is added (SRU base repair costs).

The last nodes dealt with are the base SE nodes. They have several arcs associated with them. The first arcs added are the JUMBO arcs from the set of base LRUs to the SE which are used in their repair. Next the JUMBO arcs from the set of base SRUs to the SE are added, and finally the base SE cost arc is added from the SE to the sink. Each base SE is considered separately and all the above arcs added prior to going to the next SE. The JUMBO arcs perform the same function for the base SE as they did for the depot SE, that of inter-relating the LRU/SRU sets to support the SE costs.

Minimum Cuts. Once the network is built, the set of repair level decisions are determined by applying a max-flow min-cut algorithm to identify the min-cut set. Figure 11 displays the seven potential RLA network cuts. Table IV identifies the decisions associated with these cuts and the costs incurred for each decision. Recall that a cut divides the network into two sets of nodes: one containing the source node and one containing the sink. The arcs of the cut-set are the chokepoints of the network flow. Cut-set 1 in Figure 11

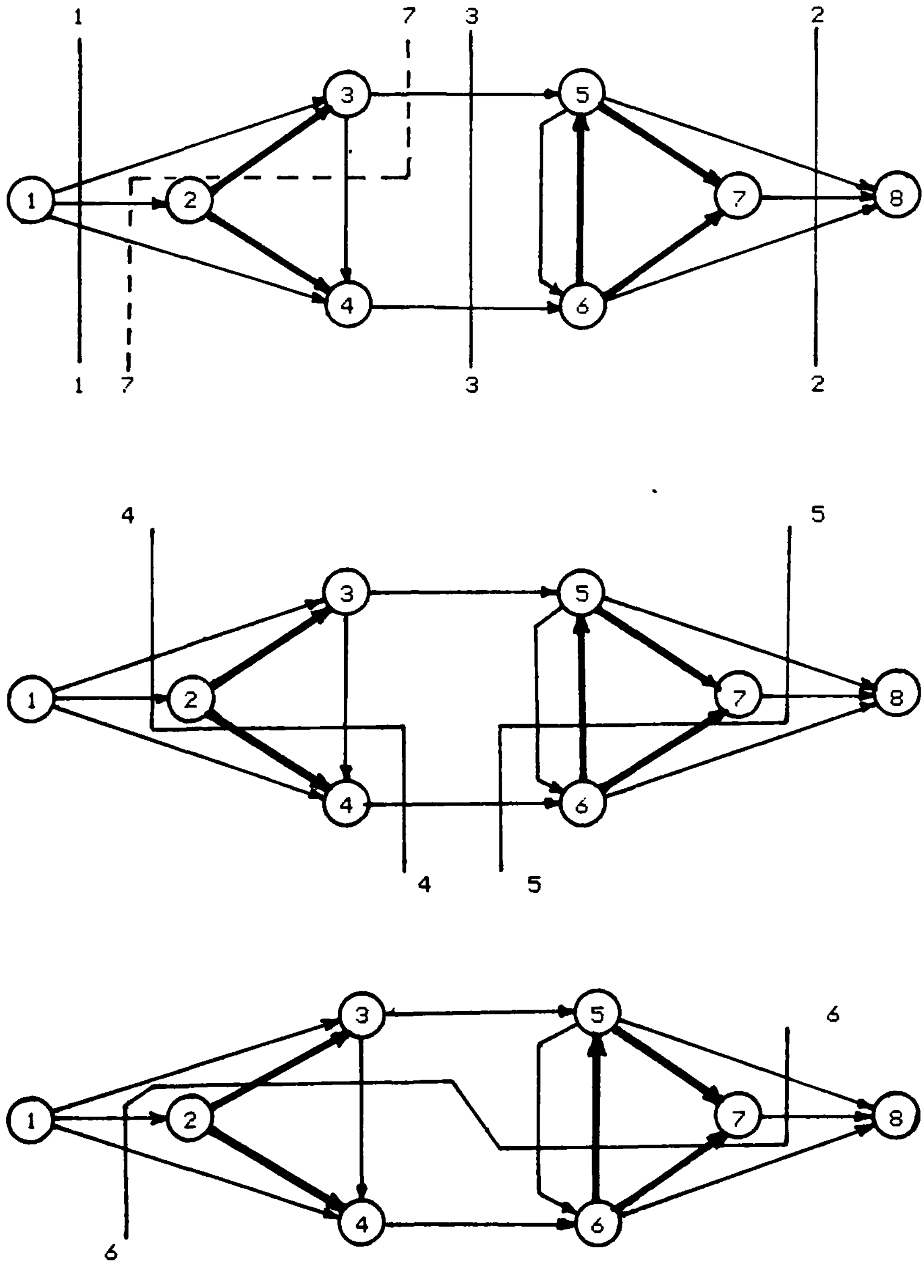


Figure 11. Potential Cuts of the RLA Network

TABLE IV

RLA Decision Costs

RLA DECISION COSTS						
Decision			Included Costs			
CUT #	LRU	SRU	SE DEP	SE BASE	DEC 1	DEC 2
1	Depot	Depot	Yes	No	No	No
2	Base	Base	No	Yes	No	No
3	Scrap	Scrap	No	No	No	No
4	Depot	Scrap	Yes	No	No	No
5	Base	Scrap	No	Yes	No	Yes
6	Base	Depot	Yes	Yes	Yes	Yes
7	Scrap	Depot	Yes	No	Yes	No

can be seen to contain the following arcs: (1,2), (1,3), and (1,4). These represent the costs associated with the depot SE, the depot LRU repair costs, and the depot SRU repair costs. This indicates that when the maximum flow was determined in the network, these arcs were capacitated; therefore, the optimal repair decisions are depot repair of both the LRU and SRU.

The following notes on the indicated cuts are necessary to ensure that the cut-sets are not misinterpreted. JUMBO arcs cannot be capacitated; therefore, all cuts crossing JUMBO arcs do not include these arcs. Cut-set 4 does not include the DEC1 arc because it originates at an unlabeled node and terminates at a labeled one; to be part of the cut-set the reverse must be true. Cut 7 represents the anomolous situation where the LRU is scrapped but the SRU is depot repaired. The cost of the SRU is included in the LRU cost, therefore a LRU scrap decision should include scrapping the SRU. The structure of the network cannot exclude the possibility of this cut occuring. If it does occur, a valid decision can be reached by running the model twice: the first time excluding the LRU scrap option, and the second time forcing the LRU and SRU scrap decisions. The optimal repair level decisions are associated with the minimum of these costs.

Now that the RLA network is built and the potential cut-sets explained, it is necessary to examine the efficiency of the algorithm used to solve the network max-flow min-cut problem. The next section covers the labeling algorithms applied to this problem.

Max-flow Min-cut Algorithmic Improvements

Conceptual Development. A primary concern expressed by the users of the NRLA model is the amount of computer resources used if the sensitivity analysis option is selected (6,9). The NRLA program allows the user to perform sensitivity analysis on a selected range of costs and/or failure rates, based on mean time between failures (MTBF), for each LRU, LRU failure mode, and SRU. This allows the user to accomplish a "what if" type analysis to determine the best repair level options in the event that one of these variables changes during the systems development. To accomplish this sensitivity analysis, the NRLA program modifies each item's cost or MTBF and then re-solves the network based on this modification. It then resets the network and performs the next modification. To get the true impact of this, consider a sample system which consists of 23 LRU failure modes and 10 SRUs with both cost and MTBF sensitivity analysis selected. In this example the sensitivity analysis program would modify and re-solve the network over 300 times. Each time the network is modified, subroutine MAXFLO is called to re-solve it. Therefore, the computational efficiency of the algorithm used in MAXFLO to solve the max-flow min-cut problem plays a major role in determining the total computer run times for the model.

Based on this information, the next step in the research effort involved analyzing the current labeling algorithm for potential time saving modifications. To assist in this analysis a detailed description of the Breadth First Labeling algorithm is presented next (17).

Breadth First Labeling Algorithm

Notation : f_{ij} = flow from node i to node j

c_{ij} = capacity from node i to node j

Concept : Assign labels to the nodes of the network such that a flow path may be determined on the network. Augment flow on these paths until the maximum flow has been attained. The labels have the form (i+,e) or (i-,e), where i is a node of the network and e is a positive integer or ∞ according to the rules specified in procedure A. The state of a node is modified by the algorithm according to whether it is unlabeled (state 1), labeled and unscanned (state 2), or labeled and scanned (state 3).

Procedure A : Start by assigning the source , s , a label of $(-,\infty)$, this indicates that it is receiving infinite flow from an external source. All other nodes are unlabeled (state 1) and the state of the source node is state 2. In general, select any node i which is in state 2. Suppose it is labeled $(k\pm , e(i))$. Scan node i for admissible arcs. Recall that admissible forward arcs require the destination node j to be unlabeled and $f_{ij} < c_{ij}$, while admissible reverse arcs require the source node j to be unlabeled and $f_{ji} > 0$. For all admissible forward arcs assign the appropriate node j the label $(i+,e(j))$ where $e(j) = \min(e(i), c_{ij}-f_{ij})$. For all admissible reverse arcs assign the appropriate node j the label $(i-,e(j))$, where $e(j) = \text{Min}(e(i),f_{ji})$. As each node is labeled a check is made to determine if the sink has been labeled. If it has proceed to Procedure B, if not continue the labeling process. If node i is completely scanned and the sink is unlabeled change node i's state to state 3 (labeled and scanned). Repeat the general step until either the sink node t is labeled, or until no additional labels can be assigned and the sink is unlabeled. If the sink is labeled go to

Procedure B, if not, stop - the maximum flow has been found.

Procedure B : The sink node t has been labeled (j+,e(t)). Trace the labeled path back to the source node s while augmenting the flow on the arcs of this path. In general, if node j is labeled (i+,e(x)), replace f_{ij} with $e(t) + f_{ij}$, and if j is labeled (i-,e(x)), replace f_{ji} with $f_{ji} - e(t)$. Proceed to node i and continue this flow augmentation until the source is reached, erase all the current labels and return to Procedure A.

Analysis of the above labeling algorithm indicates that it uses the labels it produces very inefficiently. The following simple example will more clearly illustrate this problem. Consider the network shown in Figure 12. The source node is connected to 100 nodes which are in turn connected to the sink. At each iteration of the above labeling algorithm all 100 arcs from the source to the intermediate nodes will be checked for admissibility and the nodes labeled if appropriate. After the source is completely scanned the algorithm proceed to the next labeled but unscanned node and then scans it. In this example one of two things can happen when an intermediate node is scanned; either the sink is labeled or the node is marked as scanned and a new, labeled but unscanned, node is sought. When the sink is labeled, flow is augmented on the path from the source to the sink, all the labels are erased, and the process is restarted. If the algorithm has to scan n inter-

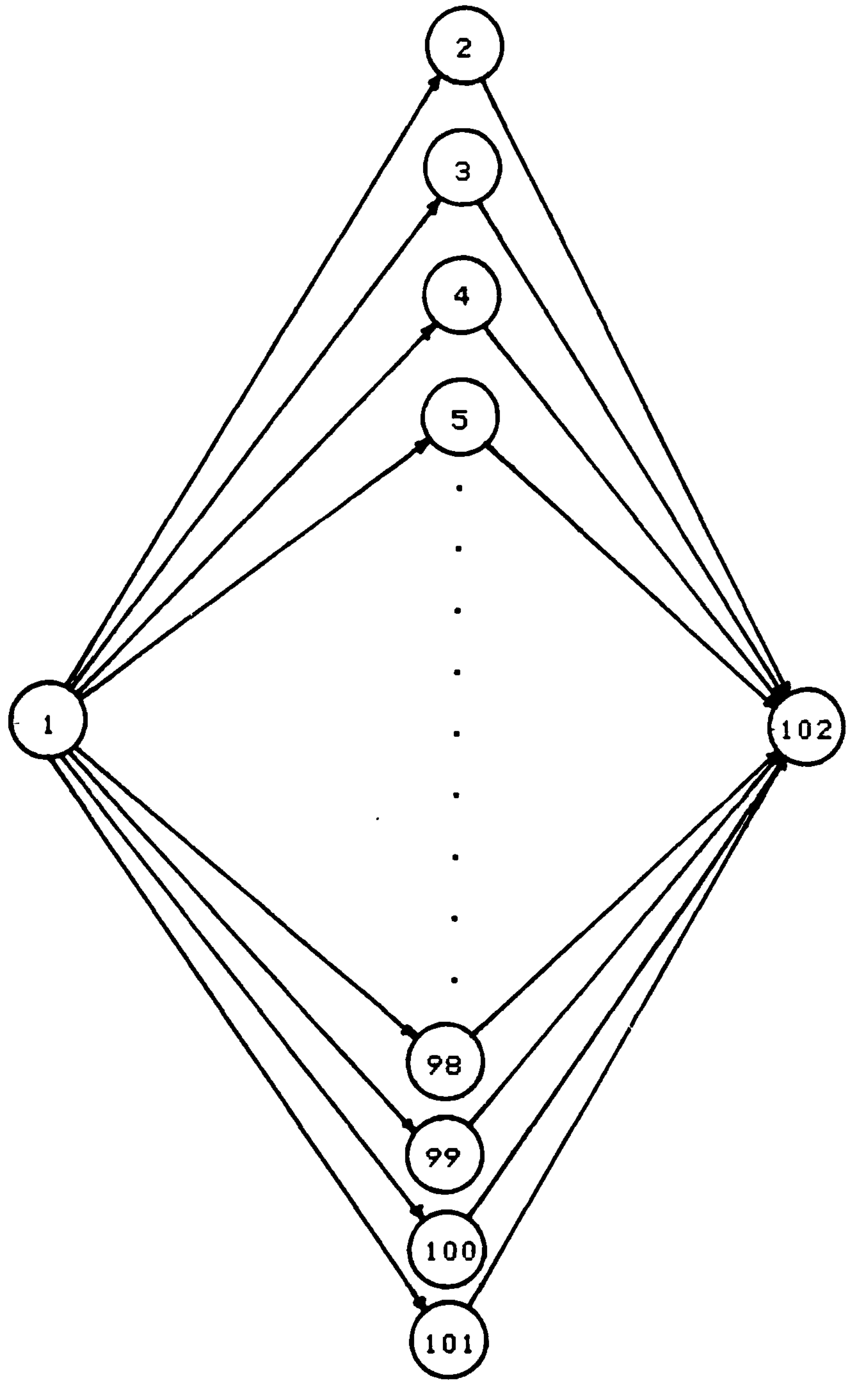


Figure 12. Network Labeling Example

mediate nodes before it finds a path to the sink, then there can be as many as 100-n nodes which are labeled and then erased without being used. Now suppose that all arcs of the network initially have flow less than capacity. Given this initial condition, for this example the algorithm would produce 99 excess labels on the first iteration, 98 on the second and so forth , for a total of 4950 excess labels. Clearly this is a less than optimal use of the labels produced.

One way to eliminate this problem involves using a Depth first search algorithm in the labeling process. A review of the literature indicated that the application of depth first search concepts to the labeling process has not been attempted before. Conceptually what a depth first search does, is move through the network from the source to the first level and then directly on to the second level and so forth until the sink is reached. A level in this context consists of the set of nodes whose shortest path from the source is the same length, where length is given in total number of arcs on the path. In the above example this would be equivalent to labeling node 2 and then immediatly labeling the sink from node 2, thus eliminating the labeling of nodes 3 through 101. RLA networks are well suited to this type search, in that only two levels separate the source and sink in most cases. The above stated advantages of a depth first search concept led to its use in developing a new labeling algorithm to solve the max-flow problem. A detailed description of this algorithm follows.

Depth First Labeling Algorithm

Notation : f_{ij} = flow from node i to node j

c_{ij} = capacity from node i to node j

Concept : Assign labels to the nodes of the network such that a flow path may be determined on the network. Augment flow on these paths until the maximum flow has been attained. Labels are assigned to the nodes on the path from the source to the sink only. Nodes are scanned only until an admissible arc is found, at which point that node is labeled and scanned for an admissible arc. The state of a node is modified according to whether it is unlabeled state 1 or labeled state 2. Nodes are not labeled as scanned or unscanned.

Procedure A : Initially all nodes are unlabeled (state 1). Proceed by labeling the source node s, $(-,\infty)$, and setting it to state 2.

First Pass (forward Paths Only)

Starting at the source check for an admissible forward arc and label its associated node i $(s+, c_{si}-f_{si})$, assign node i to state 2 and scan it for an admissible forward arc. In general proceed in this fashion until the sink is labeled or the last node on the path has no admissible forward arcs. If the sink is labeled go to Procedure B. If there are no admissible forward arcs back up to the previous node on the path and scan it for an admissible forward arc, if there is one proceed with the general rule, if not continue backing up. When no admissible forward arcs are available at the source erase all labels and go to the second pass.

Second Pass (Forward and Reverse Arcs allowed)

Once again label the source (-,∞) and set it to state 2. Scan the source for an admissible arc and label its associated node. Scan this node for an admissible arc and label its associated node according to whether it is a forward or reverse admissible arc. In general continue in this fashion until the sink is reached or no admissible arcs are available at the last node on the path. If the sink is labeled go to Procedure B. If no admissible arc is available back up to the previous node on the path and proceed. When no admissible arcs are available at the source erase all labels and proceed to the Third Pass. At this point the maximum flow has been identified.

Third Pass (labeling pass)

Repeat the procedure used for the second pass until no admissible arcs are available at the source. When this occurs stop - the max-flow with correct labels to identify the min-cut has been found.

Procedure B : The sink, node t, has been labeled (j+,e(t)). Trace the labeled path back to the source, node s, while augmenting the flow on the arcs of this path. In general, if node j is labeled (i+,e(x)), replace f_{ij} with $e(t) + f_{ij}$, and if j is labeled (i-,e(x)), replace f_{ji} with $f_{ji} - e(t)$. As the flow is augmented on this path, each node is checked to see if it has potential flow remaining. If $e(j) > e(t)$ and no previous node on the path has had potential remaining, then this node is stored as the new starting point when the flow augmentation is complete. If $e(j) \leq e(t)$ then change the nodes state to 1 and erase its label. Nodes with potential remaining, stay in state 2 and retain their labels. When the flow augmentation is complete return to the appropriate pass in procedure A and continue the pass by scanning either the stored node or the source.

Several factors contribute to the overall efficiency of the above labeling algorithm. The algorithm pushes the flow to the sink on the first available path, thereby eliminating any unused labels. It retains the position of the deepest node on the with flow potential remaining. This allows for a head start when starting the search for the next flow augmentation path. It also stores an array of the arcs on the path from the source to the sink. This eliminates the need to identify these arcs by individually scanning each node on the path. The breadth first search must identify the path in this way because the entire path is not identified until the sink is labeled.

<u>Computerization/Integration.</u> The coding of the depth first labeling algorithm was accomplished using FORTRAN IV. This version of FORTRAN was used to ensure compatibility with the existing program and to allow for transportability

between computer systems. The fully developed subroutine is called NMXFLO and is listed in Appendix F. The only part missing from this listing are the Common statements from the NRLA program. NMXFLO was designed to replace MAXFLO, and as such can be used in place of or in addition to MAXFLO in the existing NRLA model. The addition of one array (NAPATH) and two variables were the only additions to NRLAs storage requirements.

Verification/Validation. The verification of this subroutine was accomplished by developing a network data set with known arc capacities and flows. This data set was then input to NMXFLO and the resulting arc flows compared with the expected results. In this manner initial logic flaws in the algorithm were identified and corrected. When the algorithm was fully debugged and operating as anticipated it was applied to a validation phase of testing by incorporating it in the NRLA model and using it to solve the max-flow min-cut problem. The results of this testing were compared with solutions using MAXFLO (the original subroutine) for validity. In all instances NMXFLO produced identical results.

Analysis/Results. The final analysis performed on the NMXFLO subroutine dealt with determining its efficiency at solving the max-flow min-cut problem. To accomplish this, the NRLA model was run using data sets supplied by the B-1 SPO and AFALD/XRS. The execution times for the NMXFLO and MAXFLO subroutines were calculated using the CREATE computer

TABLE V

Max-flow Min-cut Algorithm Test Results

Data Set	# LRU	# SRU	# SE	# NODES	# ARCS	Processing Time in seconds (old) Breadth First	(new) Depth First	GNET
	without sensitivity analysis							
1	23	10	12	100	361	0.432	0.038	3.041
2	52	0	106	212	470	8.454	0.082	6.054
	with sensitivity analysis							
1	23	10	12	100	361	37.764	7.977	**
2	52	0	106	212	470	495.939*	15.419	**

* run time exhausted

** GNET was not compared because of the way it was implemented in the model.

system processing time program. This program clocks the actual processing time in hours. The time hacks were taken at the beginning of the subroutines and just before the return to the main program. The resulting times were then multiplied by 3600 to convert the times to computer processing seconds. The results of these tests are shown in Table V. As can be seen, the depth first search algorithm is significantly faster than the the breadth first search algorithm. The largest system tested showed the depth first algorithm with 100 fold increase in efficiency over the breadth first algorithm. The results obtained when GNET was used to solve the max-flow problem in NRLA are also contained in Table V. In this case, GNET did not overcome the efficiency of the breadth first algorithm until the system size increased to the maximum size tested, and it never approached the efficiency exhibited by the depth first algorithm. GNET was not tested for the sensitivity option due to the manner in which is was implemented in the NRLA program. The modular form in which GNET was implemented did not allow it to retain information on the network structure. This would result in it taking the total network solution time at each step of the sensitivity analysis. The other algorithms took advantage of previous information and, therefore, any comparison between them in this area would be inaccurate. Based on these results it is strongly recommended that the depth first search algorithm be incorporated as the primary MAXFLO subroutine in the current NRLA model.

VI. Centralized Intermediate Repair Facility Enhancement

Concept Development

To address the feasibility of including a CIRF maintenance option in the NRLA program required an in-depth study of the existing network formulation. The rigid structure of the existing network was not adaptive to the insertion of a fourth repair option. However, it was hypothesized that by constructing a second network, which would represent the optimal decision from the first network and the CIRF alternative, that a comparative economic analysis could be made.

To successfully implement this concept into the program, five major tasks had to be accomplished. First, a definitive description of the meaning of CIRF repair was required. Without this, it would be very difficult to validate the computerization of the CIRF repair option. The second major requirement was the identification of the necessary input data elements to support a CIRF developed network. The third objective related to the development of CIRF cost equations to calculate pipeline as well as support equipment costs. Fourthly, the various network structures that could be generated from the decisions of the first network had to be constructed, coded, and analyzed. Finally, the impact from the integration of each of these modifications had to be assessed.

The CIRF repair option is based on a generalized interpetation of how the CIRF maintenance concept is currently employed by the majority of the system program offices. Extension of the existing NRLA assumptions were made with one exception. Presently, the NRLA model assumes an uniform distribution for the basing of the total number of weapons systems being analyzed. However, in the CIRF scenario, there are additional deployment factors to be considered. Although the number of systems at each base is the same, the overriding question is the number of CIRF locations and the number of bases that are being serviced by that repair location. As an illustration, in the common situation where there are two centralized servicing facilities, one in the CONUS and one overseas, the number of bases linked to each CIRF can create an imbalanced workload. In an attempt to compensate for this imbalance, it was decided that an existing systems variable called 'OS', the fraction of the force overseas, would be applied to SE utilization requirements. The application of this variable limits the number of CIRF locations to two; one overseas and one in the CONUS. It is applied by taking the standard CIRF factors and multiplying them by the overseas fraction. These numbers are then used to calculate the SE requirements for the overseas CIRF location. The same logic is applied to the CONUS CIRF SE requirements, only now the multiplication factor is (1-OS). This results in a more accurate assessment of the true SE requirements for the CIRF alternative.

Input Data Requirements

To generate a CIRF repair option a list of the necessary CIRF variables was compiled. Once this had been done, the appropriate READ and WRITE statements were located in the program and modified to accept these additional variables. A list of these new variables and their meaning is contained in Appendix A. One particular situation that occurred during this process related to the changes that had to be made to accomodate the CIRF support equipment. Currently, the NRLA program uses the first digit of the value stored in the SECODE array to identify both the type of SE and the level at which it is used. The digits and their meaning are as follows:

1 - Common SE (Depot)
2 - Peculiar SE (Depot)
3 - Add'l SE (Depot)
4 - SE Software (Depot)
5 - Common SE (Base)
6 - Peculiar SE (Base)
7 - Add'l SE (Base)
8 - SE Software (Base)

Since the SECODE array is formatted for a four digit integer value this allowed a user to input up to 999 different types of SE in any of these classifications. To differentiate between the CIRF SE and the depot or base SE, and to also maintain the capability to have a maximum of 999 pieces of SE, the SECODE values were expanded to five digit numbers. The first two digits of the SECODE value were used for the CIRF SE as follows:

91 - Common SE (CIRF)
92 - Peculiar SE (CIRF)
93 - Add'l SE (CIRF)
94 - SE Software (CIRF)

Thus a piece of support equipment that was previously identified with the number 2013 will now be identified as 20013. If this piece of equipment needed to be used at the CIRF location its identification number would be 92013.

Cost Requirements

Each of the NRLA cost subroutines (FMCMP, LRUCMP, and SRUCMP) were expanded to incorporate a capability to compute CIRF related pipeline and SE costs. The CIRF equations were developed to be compatible with the existing depot, base, and scrap cost equations. Using the CIRF input variables, the CIRF pipeline costs were calculated based on the eleven logistical costs in Table I on page 8. Two additional cost arrays, CIRFLC and CIRFSC, were needed to store these costs for LRUs and SRUs, respectively.

To compute the CIRF SE costs, the SE requirements for each of the CIRF locations were separately determined. These requirements were then added together to obtain the total needs of the system. This value was then multiplied times the unit cost of the SE (SE array CADB) to obtain the total CIRF acquisition costs. The standard cost factor for operations and maintenance contained in SE array CODB was added to the acquisition cost along with any costs that were known to be incurred for special facilities (SE array FDB). To obtain a per base cost the summation of these costs was then divided by the total number of bases in the system. These changes are contained in subroutine SECMP in Appendix H. Having calculated the costs for a CIRF repair option, it was now necessary to design the correct RLA networks for the CIRF option.

CIRF Network Conceptual Development

Initially, the incorporation of CIRF into the NRLA model was visualized as requiring only a simple modification of the network structure to include the CIRF option. However, this proved to be false. A major restriction on tnis modification was that it had to be accomplished without loosing the ability to identify the optimal repair level decisions through the min-cut set. Upon investigation, it became apparent that the inclusion of CIRF in this manner could not be accomplished. There is no network structure that retains the ability to identify a unique repair level decision from the four repair options. This is because of the unique use of a maximal flow algorithm to solve the minimal cost problem.

The next option considered, involved exploring the feasibility of making the decisions in a two-pass scenario. The initial pass would compare the standard depot-scrap-base configuration and the second pass would then compare these optimal decisions against the CIRF option. This method of incorporating CIRF proved to be feasible and was developed for inclusion in NRLA.

An initial problem to be overcome with this method was how to handle the seven potential SRU/LRU decision pairs which can result from the first pass. In each instance the network generating subroutine must identify the SRU/LRU decision pair and build an appropriate network structure so that the resulting comparison with CIRF will provide valid repair level decisions. The potential for seven differently constructed subnetworks brings up the problem of how to identify the optimal decisions in these different structures. The solution to these problems led to the development of the set of seven subnetworks shown in Figures 13(a) thru 13(g). As can be seen in these figures, the basic RLA network structure is retained for each subnetwork. One major difference is that both the depot and base SE are now included on the right-hand side of the network structures.

CIRF Network Construction. Constructing the networks displayed in Figure 13 required a major modification of the SETNET subroutine. The manner in which the nodes were assigned changed very little. The source is assigned first, followed by the CIRF SE, then the CIRF LRUs and SRUs. These nodes replace the depot equivalents in the original RLA networks. The next nodes assigned are the LRU and SRU nodes associated with the optimal decisions from the first NRLA pass. These nodes take the place of the base LRU and SRU nodes in the original RLA networks. Next the depot SE nodes are assigned, followed by the base SE nodes, and then the sink node. This structure requires that the total number of nodes be increased by the number of CIRF SE.

The manner in which the arcs are connected to these nodes is the key ingredient in obtaining the optimal repair level decisions. The arcs originating at the source and CIRF SE

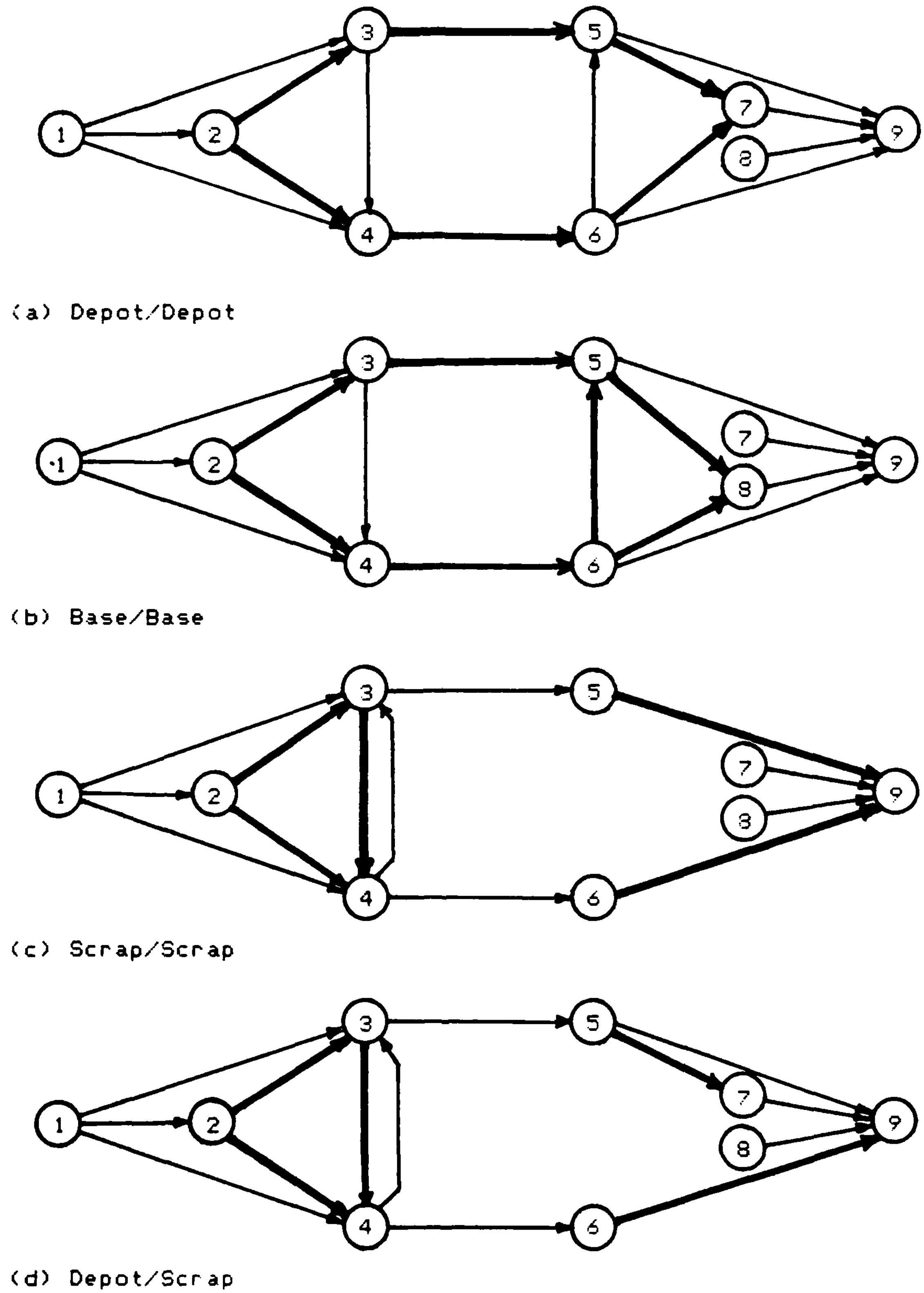
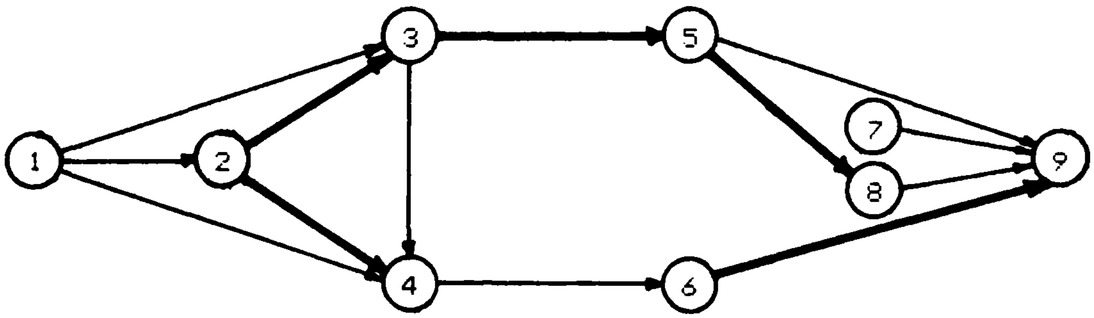
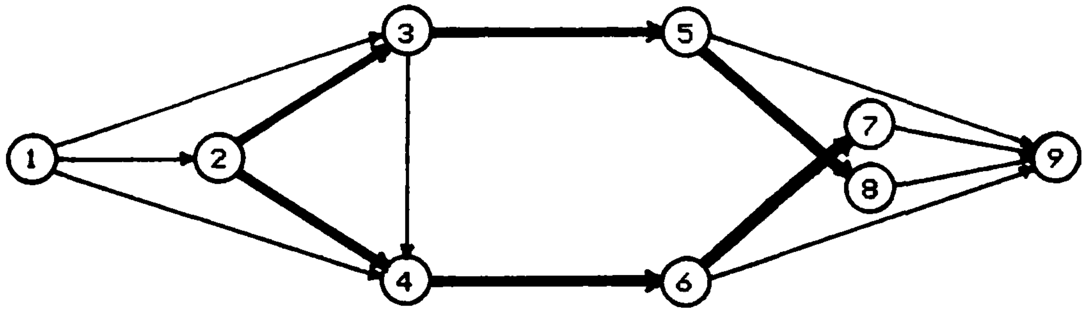


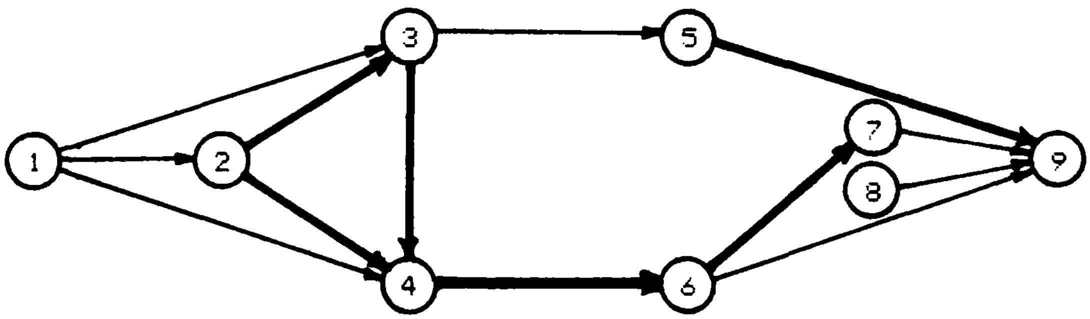
Figure 13 (a,b,c,d) CIRF RLA Subnetworks



(e) Base/Scrap



(f) Base/Depot



(g) Scrap/Depot

Figure 13 (e,f,g) CIRF RLA Subnetworks

are the same as those in the original RLA network. The first arcs requiring special attention are those that originate at the CIRF LRUs.

The arcs originating from the CIRF LRUs are assigned in the following order. First the arc connecting the LRU to the associated CIRF SRU, and then the arc connecting the LRU to the NRLA decision LRU. The first arcs capacity will be DEC 3 if the LRU decision from the first pass is base repair, and JUMBO otherwise. The DEC 3 costs are those pipeline costs associated with base repair of the LRU and CIRF repair of the SRU. The JUMBO capacity is used to exclude undesirable decisions such as scrapping or depot repairing the LRU and CIRF repairing the SRU. The second arc is associated with the scrap decision for the LRU and as such will have a capacity of JUMBO if the first pass decision was not scrap. If the decision was scrap, then the the capacity of this arc is the cost associated with the scrap decision.

The CIRF SRU arcs are assigned next. The first arc connects the SRU to the CIRF LRU node. The capacity of this arc is DEC 2 and it corresponds to the pipeline costs of replacing the SRU when the LRU is CIRF repaired and the SRU is scrapped. The next arc originating from the CIRF SRU is connected to the NRLA SRU decision node. As with the LRU above, this arc's capacity will be JUMBO if the SRU decision was not scrap. If the decision was scrap, then a check is made to determine which decision was made for the LRU. If the LRU decision was base repair, then the DEC 2 costs are added to the SRU scrap cost and this is used as the arc capacity. If the LRU decision was not base, then the SRU scrap cost alone is used as the arcs capacity.

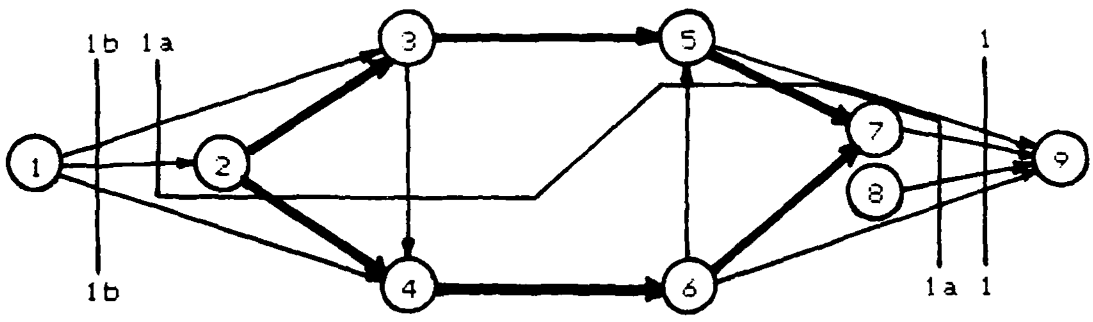
The next arcs entered are those associated with the NRLA LRU decisions. The first arc connects the LRU node to the sink. If the NRLA decision was scrap, then the capacity of this arc is JUMBO. If it was base or depot, then the corresponding LRU repair cost is used as its capacity. Following this, the JUMBO capacity arcs connecting the LRU to the SE are assigned. The only arcs assigned, connect the LRU to the SE associated with the NRLA LRU repair decisions. As an example, if the LRU decision was base repair, then the arcs connecting the LRU node to the base SE would be assigned.

The NRLA SRU arcs are assigned next. The first arc assigned connects the NRLA SRU to the NRLA LRU. This arc is assigned in only two instances; (1) if the NRLA decisions were base repair of both the SRU and LRU and (2) if the NRLA decisions were depot repair of both. When the decisions are base repair, the arc capacity is JUMBO. This ensures that the SRU is not base repaired when the LRU is CIRF repaired. When the decision is depot repair of both, then the arc capacity is the sum of the DEC 1 and DEC 2 costs. This adds the appropriate DEC costs for CIRF LRU repair and depot SRU repair. Following this, the arcs to the sink are added. The capacity of these arcs are JUMBO if the NRLA decision was scrap the SRU. If both the LRU and SRU are repaired at the same facility the corresponding SRU repair cost is used as the capacity for this arc. In the instances where the LRU is decisions are scrap or base repair and the SRU decision is depot repair, then the arc capacity is the sum of the SRU depot repair cost plus the DEC 1 and DEC 2 costs. The last arcs added that originate from the NRLA SRUs are the JUMBO capacity arcs to the appropriate SE resources.

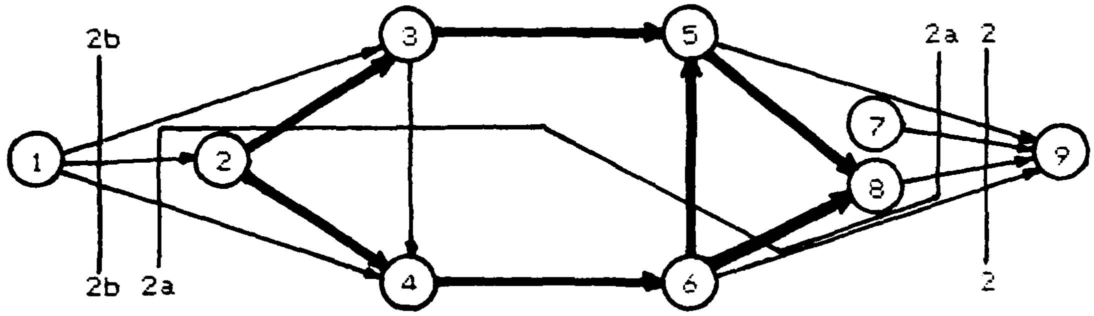
The first arcs added to the network connect the base and depot SE resources to the sink. The capacity of these arcs are the SE costs.

Although the above set of network structures is fairly complicated, it does provide a unique set of repair level decisions for the DEPOT-SCRAP-BASE-CIRF repair options. A vital element of this structure is the standardization of the CIRF repair nodes and arcs on the left side of the network. This allows the output subroutine to identify changes in the repair level decisions by just checking the state of the CIRF LRU and SRU nodes. If these nodes are in state 1 (unlabeled), then the LRU or SRU optimal repair level decision is CIRF. If the nodes are in state 2 (labeled) then the original NRLA decision is optimal.

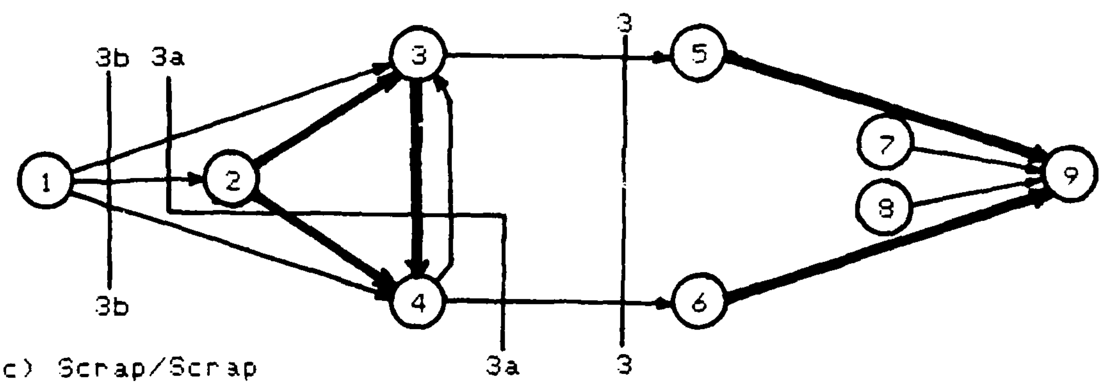
CIRF Network Cut Sets. The differences in the structures of the subnetworks lead to different cut-sets for each structure. Figures 14(a) thru 14(g) depict the feasible cut-sets possible. Table V presents the costs associated with these cuts. The same restrictions apply to these cut sets that applied to the original NRLA cut sets. The JUMBO capacity arcs cannot be part of the cut set, and the cut arcs must originate at a labeled node and terminate at an unlabeled one.



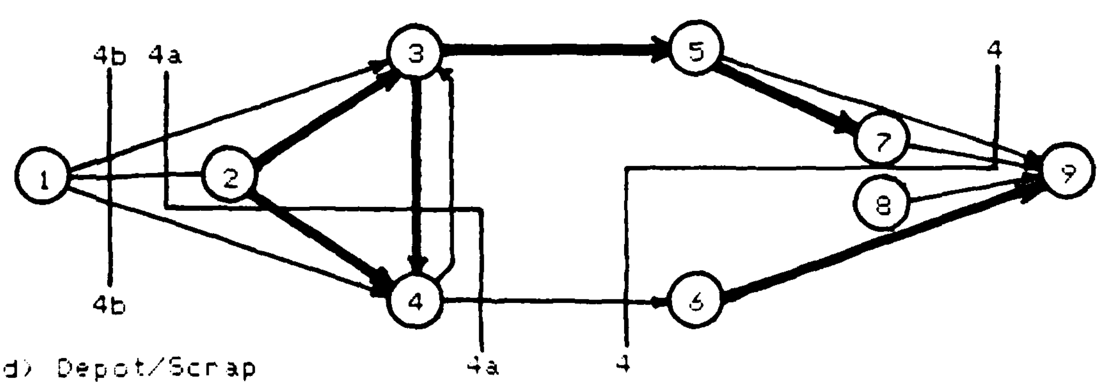
(a) Depot/Depot



(b) Base/Base

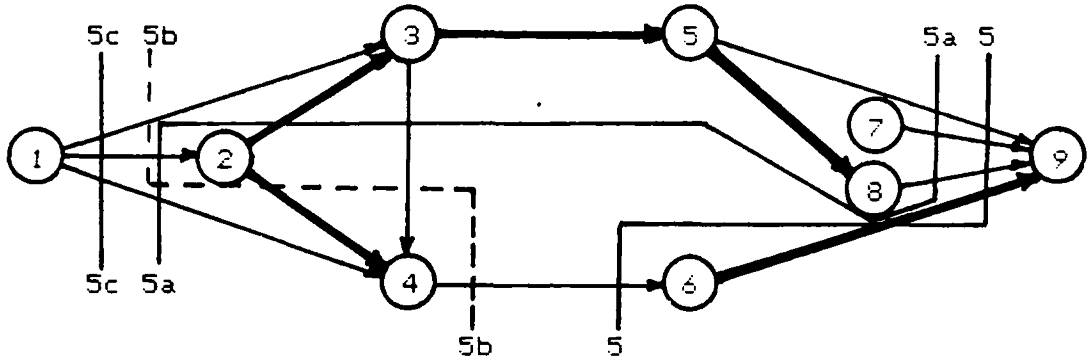


(c) Scrap/Scrap

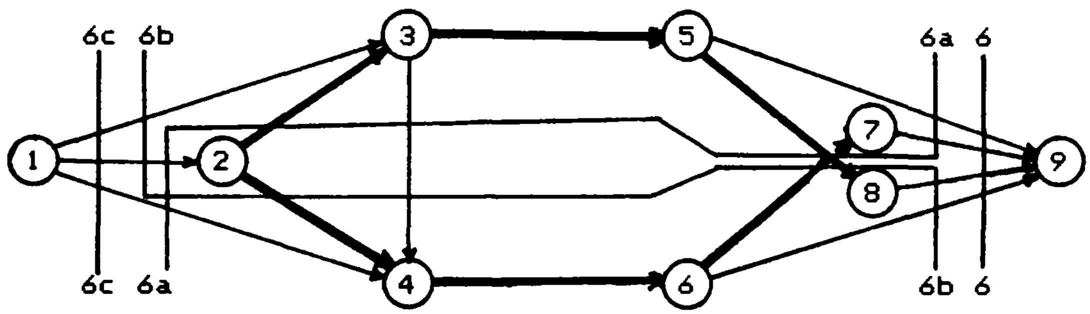


(d) Depot/Scrap

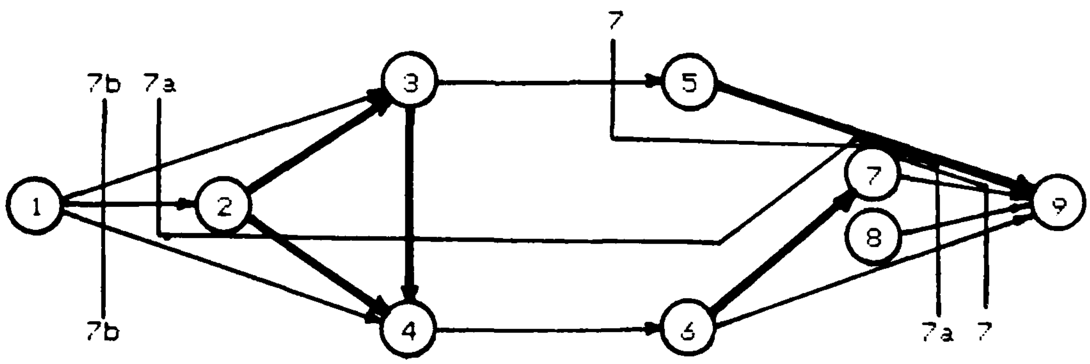
Figure 14 (a,b,c,d) CIRF RLA Network Potential Cuts



(e) Base/Scrap



(f) Base/Depot



(g) Scrap/Depot

Figure 14 (e,f,g) CIRF RLA Network Potential Cuts

TABLE VI

CIRF Network Decision Costs								
	Decision		Included Costs					
	LRU	SRU	DEP	SE BASE	CIRF	DEC 1	DEC 2	DEC 3
1	Depot	Depot	X					
1.a	CIRF	Depot	X		X	X	X	
1.b	CIRF	CIRF			X			
2	Base	Base		X				
2.a	Base	CIRF		X	X			X
2.b	CIRF	CIRF			X			
3	Scrap	Scrap						
3.a	CIRF	Scrap			X		X	
3.b	CIRF	CIRF			X			
4	Depot	Scrap		X				
4.a	CIRF	Scrap			X		X	
4.b	CIRF	CIRF			X			
5	Base	Scrap		X			X	
5.a	Base	CIRF		X	X			X
5.b	CIRF	Scrap			X		X	
5.c	CIRF	CIRF			X			
6	Base	Depot	X	X		X	X	
6.a	Base	CIRF		X	X			X
6.b	CIRF	Depot	X		X	X	X	
6.c	CIRF	CIRF			X			
7	Scrap	Depot	X			X		
7.a	CIRF	Depot	X		X	X	X	
7.b	CIRF	CIRF			X			

Computerization/Integration

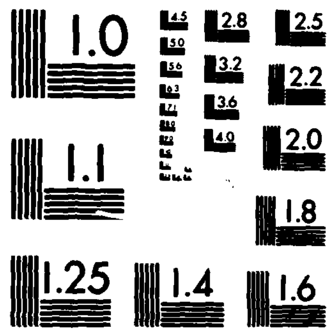
The inclusion of a CIRF capability in the NRLA program required extensive changes to the program. Subroutines OUTPUT and SECMP were modified to accept and work with the new SE codes. These subroutines also identify when CIRF is included in the analysis so that the appropriate calculations can be performed. The extensive reports prepared by NRLA were a significant hinderance in achieving complete program integration. The code listing for OUTPUT and SECMP are contained in Appendixes G and H. Subroutine CRFSET creates the new network structures and a listing of it is contained in Appendix C. Once again all changes and code were accomplished using FORTRAN IV for compatibility. In addition to the above major changes, numerous smaller changes were made in subroutines LRUCMP, FMCMP, and SRUCMP. The MAIN program also required changes to incorporate CIRF. This portion of the program also has a significant report writing capability which required modification.

Verification/Validation

The verification of subroutine CRFSET, which builds the CIRF networks, was accomplished in two stages. The first stage involved the development of a set of network data which required the subroutine to build all possible network structures. The second stage of the verification required running the CRFSET subroutine in conjunction with SETNET and NMXFLO to ensure that the input/output interphases worked as expected and to further identify any potential problem[illegible]

AD-A141 198 ENHANCEMENTS TO THE NETWORK REPAIR LEVEL ANALYSIS (NRLA) MODEL USING MARG..(U) AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOOL OF ENGI.. 2/2

UNCLASSIFIED G W ARNETT ET AL. DEC 83 AFIT/GOR/OS/83D-3 F/G 15/5 NL



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

The data set developed for this test included all possible cut set combinations so that all potential logic errors could be identified.

The data set developed for the first stage was designed to build all seven possible network structures. The arcs in these seven structures were assigned capacities equal to their arc number. The arc numbers were based on the expected arc assignments from the developed program logic. This set was then read into a test program and CRFSET was called and implemented. The resulting network structures were then compared with the expected results for accuracy of execution. When it was verified that the program was building the networks correctly the second stage of testing began.

The second stage required three data sets to test the entire series of cut set combinations. During this test the data sets were read into the test program, and processed by SETNET to build the original RLA network. This network was then solved by NMXFLO and the results printed out and checked to ensure that the input data set was producing the expected set of repair level decisions for input to CRFSET. Following this, CRFSET was called to build the CIRF network structures. These were then input to NMXFLO for solution. The results of the network max-flow solution was printed out and compared with the projected results. The results of these tests conclusively proved that subroutine CRFSET was processing the input data and producing the correct network structure in all cases. The final task dealing with the CIRF capability, was to integrate it into the NRLA program. This task proved to be much more complicated than initially anticipated.

There were two primary factors which increased the complexity of the integration project. The first factor was the extensive work required to update the report capabilities of the program to include CIRF. This included changing much of the logic of the code so that the new SE codes could be accepted. The second factor which is directly tied to this effort is the user friendliness of the CREATE diagnostic system. On several occasions during the debugging of the program it was noted that the diagnostic outputs from the CREATE system shed little light on the nature of the problem encountered. Logic error reports were usually obscure or non-existent. In one instance an entire subroutine was missing from the program being tested, yet no recognizable diagnostic error showed up to indicate this deficiency.

Validation of the CIRF repair level option requires final integration of CIRF into the NRLA model. In addition, the cost equations developed for CIRF repair must be validated by the appropriate cost analysts in AFALC.

Analysis/Results

The final analysis of the CIRF capability requires total integration into the model and validation of the cost equations developed. The capability to include CIRF in the RLA networks is completely verified and will provide a significant analytical tool for logistics analysts when it has been integrated into the NRLA model.

VII. Conclusions/Recommendations

General Comments.

The NRLA program is a unique application of networking to solve the repair level analysis problem. It is extensively used by the Air Force logistics community to aid in determining the optimal repair level decisions for the components of new systems being procured by the Air Force. It is anticipated that models of this type will continue to play an increasingly important role in this process and as such, improvements to these models provide a significant benefit to the Air Force.

During the course of this thesis effort several areas of the NRLA program were identified as being candidates for further study and modification. In general, it is felt that the current implementation of the program should be rewritten using a more current version of FORTRAN. This will provide significant benefits in the versatility of the programming techniques available for further enhancements to the model. The sensitivity analysis section of the model requires study to determine if it can be implemented with the CIRF capability. The data structures of the model provide an area for future enhancements using new and more efficient techniques such as those found in Jensen and Barnes (4). Modification of the data structures will require a major revision of the model and, therefore, should provide an excellent opportunity to upgrade the model to the newer version of FORTRAN as recommended above.

Recommendations

MARLA. Although the MARLA enhancement has been verified and integrated into the model, conflicting repair decisions were experienced. It appears that these inconsistencies can be resolved with further effort. Based on the possible efficiencies that could be realized from the use of MARLA as a preprocessor to the NRLA model, it is recommended that this effort be undertaken. This recommendation is compatible with the Air Force guidance as outlined in AFLC/AFSC Pamphlet 800-4 (13). Following this, MARLA should be fully tested by the users to determine the extent of the benefits available from its use. For very large systems MARLA may be the best tool available to reduce the size of the data input to NRLA. In any case MARLA can be utilized as a separate program to perform marginal analysis by developing it as a program for use on micro computer systems.

Labeling Improvements. The Depth First Search Labeling algorithm developed for use in the NMXFLO subroutine is an original variation of currently accepted networking algorithms. Based on the results achieved during this study a high level of confidence was developed in its ability to outperform the existing procedure as well as other alternative methods. From this perspective it is recommended that this enhancement be immediately implemented for use in the NRLA model and distributed to current users of the model as a modular replacement for the MAXFLO subroutine. The use of this subroutine should significantly reduce the computer resources expended during NRLA analysis, thus saving the Air Force time and money.

CIRF The ability to analyze a CIRF repair option is a sorely needed capability in the logistics planning area. By using a new series of network structures an answer has been formulated to this problem. Verification of this effort has been completed and full integration into the NRLA program is recommended at this time. A special consideration related to the CIRF effort is the validation of the cost equations by qualified cost analysts in AFALC.

Conclusion.

In summary, for the integrated logistics support process to be successful, requires that repair level analysis be skillfully performed. Without question, the problem solving technique employed in the NRLA model produces economically based optimal repair level recommendations. However, it should be remembered that the validity of these computer generated decisions is directly dependent upon the accuracy of the data input and the cost estimating relationships used.

By accomplishing the stated thesis objectives, substantial progress has been achieved in increasing both the efficiency and the capabilities of the NRLA model. Because of these improvements, it is now appropriate to investigate methods to incorporate the other non-economic factors which influence the repair level decision. A possible approach could be the development of an objective function consisting of the elements (economic and non-economic) which are identified as critical to the systems performance. This fuctional relationship could then be used to establish the appropriate "cost" for each arc's capacity in the network. In this way such factors as a system's mobility and deployment requirements, availability of skilled maintenance personnel, or operational readiness requirements could be considered as part of the analytical optimization process used by NRLA. Proper integration of these effects would better reflect the total dynamics of the multi-item, multi-indenture, multi-level system which has been studied during this effort. If these recommendations are implemented a more responsive and versatile tool will be available to Air Force decision makers in the planning and management of the logistical support structures necessary for tomorrow's weapons systems.

Appendix A: Glossary of Variables

CAA - available work time per month for a centralized intermediate level maintenance man (man-hours/month, Maintenance System Data Record).

CIRFLC - an array of values, one for each LRU failure mode, each of which is the sum of certain life cycle logistics costs associated with intermediate level repair of the LRU.

CIRFSC - an array of values, one for each SRU, each of which is the sum of certain life cycle logistics costs associated with intermediate level repair of the SRU.

CLR - hourly labor rate for centralized intermediate level maintenance man ($/hour, Maintenance System Data Record).

CMMH - the number of maintenance man-hours required for repair of an LRU if the repair is done at CIRF level (man-hours/repair, LRU Failure Mode Data Record).

CMMHS - the number of maintenance man-hours required for repair of an SRU if the repair is done at the CIRF level (man-hours/repair, LRU Failure Mode Data Record.)

CRCTC - the elapsed time from removal of a failed LRU at CONUS CIRF until the item could become a serviceable spare in depot stock, it includes the time required for base to depot transportation and the depot shop flow time required for repair (months, LRU Data Record).

CRCTO - the elapsed time from removal of a failed LRU at an overseas CIRF until the item could become a serviceable spare in depot stock (months, LRU Data Record).

CRCTPL - the expected number of unserviceable LRU assets in the CIRF repair pipeline (No.LRU,Computed).

CRCTSL - the number of spare LRUs to be purchased to satisfy LRU demands expected to occur during the CIRF repair cycle time (No. LRUs,Computed).

HRAVSE - an array for each item of support equipment which stores the available equipment hours.

NAPATH - an array which is used to store the arcs on the path from the source to the sink.

NTCLFM - an array of values, one for each LRU failure mode, each of which is the minimum number of CIRF level maintenance personnel to be trained for the LRU repair task (number of people, LRU Failure Mode Data Record).

NTCS - an array of values, one for each SRU, each of which is the minimum nimber of CIRF level maintenance personnel to be trained for the SRU repair task (number of people, SRU Data Record)

SCRCTC - SRU CIRF repair cycle time for CONUS bases; the elapsed time from removal of a failed SRU (from the LRU) at a CONUS base until the item could become a serviceable spare in depot stock (months, SRU Data Records).

SCRCTL - the number of spare SRUs to be purchased to satisfy SRU demands expected to occur during the SRU CIRF repair cycle time (No.SRU,Computed).

SCRCTO - SRU CIRf repair cycle time for overseas bases; the elapsed time from removal of a failed SRU from the LRU at an overseas base until the item could become a serviceable spare in depot stock (months,SRU Data Record).

SCRCTP - the expected number of unserviceable SRU assets in the CIRF repair pipeline (No.SRU,Computed).

SEARY - a support equipment/LRU/SRU cross reference array, which stores the pointers of the SE for a particular failure mode or SRU.

TBCST - a cost array which collects the total cost of repair at the base level for an LRU or SRU.

TCCST - a cost array which collects the total cost of repair at the CIRF for an LRU or SRU.

TDCST - a cost array which collects the total cost of repair at the depot level for an LRU or SRU.

TFD - training factor for depot; the expected number of times that formal maintenance training will be required for depot personnel (dimensionless, Computed).

TLCD - total life cycle repair demands for an LRU at each base (No.repair demands,/L.C.,Computed).

TLCDF - total life cycle repair demands for a particular failure mode of an LRU at each base (No.repair demands,/L.C., Computed).

TRB - annual turnover rate for intermediate level maintenance personnel (fraction of personnel replaced/year, Maintenance System Data Record).

TRC - the expected training cost, instruction and materials, for LRU repairs ($/Man/week, LRU Failure Mode Data Record).

TSECB - a cost array used to collect the SE costs at the base level for a particular LRU or SRU.

TSECC - a cost array used to collect the SE costs at the CIRF for a particular LRU or SRU.

TSECD - a cost array used to collect the SE costs at the depot level for a particular LRU or SRU.

TSECST - the total cost associated with the acquisition and operation of a particular piece of support equipment.

Appendix B: NRLA Subroutine Descriptions

DECIDE - this subroutine is called from UCLSA and MTBFSA. Its purpose is to specifically determine the decision changes identified by the sensitivity analysis routines and document the changes on an output file.

FMCMP - this subroutine is called by MAIN, MTBFSA, and UCLSA to compute repair level option costs for an LRU failure mode. These costs are saved in arrays DEPOLC, SCRPLC, CIRFLC, and BASELC. The expected number of maintenance man-hours required monthly for the failure mode is computed for depot, CIRF, and base and saved in FSEUHD, FSEUHC, and FSEUHB respectively.

LRUCMP - this subroutine is called by MAIN, MTBFSA and UCLSA. Its function is to compute inventory stock levels and life cycle SCRAP option costs for an LRU.

MAXFLO - this subroutine is called by MAIN, MTBFSA, and UCLSA. Its purpose is to determine the maximum flow through the network and the minimum cut set. This cut set identifies the minimum cost set of repair level decisions.

MTBFSA - this subroutine is called from MAIN to determine the effects of changes in the MTBF for an LRU. The effects could be changes in failure mode repair level decisions, SRU repair level decisions, and/or changes to SE decisions.

OUTPUT - this subroutine is called from MAIN to print the optimal solution results. This includes support equipment decisions plus LRU and SRU repair level decsisions.

RESET - this subroutine is called by MAIN, MTBFSA, and UCLSA. Its purpose is to exploit the known structure of the RLA network so that advance flow can be placed on the network arcs to reduce the solution time for the MAXFLO subroutine.

SECMP - this subroutine is called by MAIN and MTBFSA. It determines the quantity of each SE resource potentially required at depot, CIRF, and base level plus the life cycle cost for the resources.

SETNET - this subroutine is called by MAIN after all costs and dependency relationships are known. The function of SETNET is to use this information to construct a matrix representation of the RLA network, determine and save pointers for use by the solution algorithm and save pointers for use by·the sensitivity analysis subroutines.

SORT - this subroutine is called from SETNET. As ists name implies, its function is to sort values supplied by SETNET. Specifications to SORT are contained in its arguments list.

SRUCMP - this subroutine is called by MAIN, MTBFSA, and UCLSA to compute SRU related life cycle costs. These costs are saved in the SRU related arrays DEPOSC, SCRPSC, CIRFSC, BASESC, BRLSS, BRLDS, and BRLCS.

UCLSA - this subroutine is called by MAIN to determine the effects of LRU and SRU unit cost changes. The routine determines if decision changes will occur over a range of values for both the LRU's and SRU's unit cost.

Appendix C: CRFSET Subroutine

```
      SUBROUTINE CRFSET(MSGL2)
C
C
      ICIRF=2
C
C      CODE FROM STATEMENT # 2000 THROUGH # 2210 UTILIZES THE ABOVE
C      COMPUTED ITEM & SE COSTS TO PRODUCE AN RLA COST NETWORK.
C
C      AS A PRELIMINARY STEP DETERMINE NODE NUMBERS FOR KEY NODES IN
C       THE NETWORK.  THE VARIABLES ARE CODED WITH THE PREFIXES 'LC'
C       FOR 'LAST CIRF' AND 'LN' FOR 'LAST NRLA INPUT'
C
 2000 LCSE=NCSE+1
      LCLRU=LCSE+LFMS
      LCSRU=LCLRU+SFMS
      LNLRU=LCSRU+LFMS
      LNSRU=LNLRU+SFMS
      LNDSE=LNSRU+NDSE
      LNBSE=LNDSE+NBSE
      LNODE=LNBSE+1
      IF(LNODE.LE.MAXNOD)  GO TO 2005
      WRITE(6,2002)
 2002 FORMAT('0 NODE VECTORS ARE TOO SMALL -- STOP')
      STOP
C
C
C       SET # OF ARCS TO ZERO & # OF NODES TO 1
 2005 NARCS=0
      NNODES=1
      NBSEP0=NDSE+NBSE
      NBSEP1=NDSE+NBSE+1
C
      IF(NCSE.LT.1)  GO TO 2015
C       CREATE ARCS FROM SOURCE TO CIRF SE -- CIRF SE COST ARCS
      DO 2010 I2010=NBSEP1,NUMSER
      NARCS=NARCS+1
      NNODES=NNODES+1
      SRCE(NARCS)=1
      DEST(NARCS)=NNODES
      CAP(NARCS)=SECOST(I2010)
 2010 CONTINUE
C
C       CREATE ARCS FROM THE SOURCE TO THE CIRF LRU FAILURE MODE
C       NODES -- ITEM RELATED COSTS FOR LRU REPAIR AT CIRF
 2015 DO 2020 I2020=1,LFMS
      NARCS=NARCS+1
      NNODES=NNODES+1
      SRCE(NARCS)=1
      DEST(NARCS)=NNODES
      CAP(NARCS)=CIRFLC(I2020)
```

```
 2020 CONTINUE
C
C      CREATE ARCS FROM THE SOURCE TO THE CIRF SRU NODES -- ITEM
C      RELATED COSTS FOR SRU REPAIR AT CIRF
      IF(SFMS.LT.1)  GO TO 2040
      DO 2030 I2030=1,SFMS
      NARCS=NARCS+1
      NNODES=NNODES+1
      SRCE(NARCS)=1
      DEST(NARCS)=NNODES
      CAP(NARCS)=CIRFSC(I2030)
 2030 CONTINUE
C
C      CREATE ARCS FROM THE CIRF SE NODES TO THE CIRF LRU/SRU NODES
 2040 IF(NCSE.LT.1)  GO TO 2070
C      FOR EACH CIRF SE --
      DO 2060 I2060=NBSEP1,NUMSER
C      SEPF CONTAINS A POINTER TO THE SE CROSS REFERENCE (SEXREF)
      IPTR=SEPF(I2060)
      IF(IPTR.EQ.0)  GO TO 2060
C      SEXREF CONTAINS THE # OF AN LRU FAILURE MODE (ITEM > 0) OR
C      THE NUMBER OF AN SRU (ITEM < 0)
 2050 ITEM=SEXREF(IPTR)
      IF(ITEM.LT.0)  ITEM=LFMS-ITEM
      NARCS=NARCS+1
      SRCE(NARCS)=I2060-NBSEP0+1
C      ITEM IS A POINTER TO AN LRU OR SRU AND WHEN ADDED TO LCSE
C      GIVES THE APPROPRIATE NODE NUMBER
      DEST(NARCS)=LCSE+ITEM
      CAP(NARCS)=JUMBO
C
C      NXTITM IS A POINTER TO THE NEXT ENTRY IN SEXREF WHICH IS AN
C      LRU OR SRU REQUIRING THE CURRENT SE, I.E., SE # I2060.
      NPTR=NXTITM(IPTR)
      IF(NPTR.EQ.0)  GO TO 2060
      IPTR=NPTR
      GO TO 2050
 2060 CONTINUE
C
C      CREATE THE ARCS EMANATING FROM THE CIRF LRU FAILURE MODE NODES
 2070 DO 2080 I2080=1,LFMS
C      CHECK FOR AN SRU
      IPTR=SRUPTR(I2080)
      IF(IPTR.EQ.0)  GO TO 2075
C      CREATE AN ARC TO THE CIRF SRU NODE -- COSTS UNIQUE TO BASE
C      REPAIR OF LRU & CIRF REPAIR OF SRU
      NARCS=NARCS+1
      SRCE(NARCS)=LCSE+I2080
      DEST(NARCS)=LCLRU+IPTR
       CAP(NARCS)=BRLCS(IPTR)
       IF (OPDECL(I2080).NE.LOCAT(3)) CAP(NARCS)=JUMBO
C       CREATE AN ARC TO THE NRLA LRU NODES -- COST OF SCRAPPING THE LRU
```

```
2075 NARCS=NARCS+1
     SRCE(NARCS)=LCSE+I2080
     DEST(NARCS)=LCSRU+I2080
     CAP(NARCS)=SCRPLC(I2080)
     IF (OPDECL(I2080).NE.LOCAT(2)) CAP(NARCS)=JUMBO
2080 CONTINUE
C
C      CREATE ARCS FROM THE CIRF SRU NODES TO THE NRLA SRU NODES --
C      COST OF SCRAPPING THE SRU
     IF(SFMS.LT.1)  GO TO 2100
     DO 2090 I2090=1,SFMS
C      CREATE ARCS FROM THE CIRF SRU NODES TO THE CIRF LRU
C      NODES, TO ELIMINATE INFEASIBLE LRU/SRU REPAIR LEVEL
C      MATCHES SUCH AS LRU-D, SRU-C; LRU-S, SRU-C; LRU-C, SRU-B
     IF (OPDECS(I2090).EQ.LOCAT(2).AND.OPDECL(LRUPTR(I2090)).
    &EQ.LOCAT(1)) GOTO 2082
     IF (OPDECS(I2090).EQ.LOCAT(2).AND.OPDECL(LRUPTR(I2090)).EQ.
    &LOCAT(2)) GOTO 2082
     GOTO 2085
2082 NARCS=NARCS+1
     SRCE(NARCS)=LCLRU+I2090
     DEST(NARCS)=LCSE+LRUPTR(I2090)
     CAP(NARCS)=BRLSS(I2090)
C      CREATE ARCS FROM CIRF SRU TO NRLA SRU -- SCRAP COSTS
2085 NARCS=NARCS+1
     SRCE(NARCS)=LCLRU+I2090
     DEST(NARCS)=LNLRU+I2090
     CAP(NARCS)=SCRPSC(I2090)
     IF (OPDECS(I2090).NE.LOCAT(2)) CAP(NARCS)=JUMBO
C      ADD DEC2 COSTS TO SCRAP CSOT FOR BASE SCRAP DECISION
     IF (OPDECL(LRUPTR(I2090)).EQ.LOCAT(3).AND.OPDECS(I2090).EQ.
    &LOCAT(2)) CAP(NARCS)=CAP(NARCS)+BRLSS(I2090)
2090 CONTINUE
C
C      CREATE ARCS EMANATING FROM THE NRLA LRU NODES
2100 DO 2120 I2120=1,LFMS
C      CREATE AN ARC FROM THE NRLA LRU NODE TO THE LAST NOD (SINK) --
C      ITEM RELATED COST OF NRLA REPAIR OF THE LRU
2110 NARCS=NARCS+1
     SRCE(NARCS)=LCSRU+I2120
     DEST(NARCS)=LNODE
     CAP(NARCS)=JUMBO
     IF (OPDECL(I2120).EQ.LOCAT(3)) CAP(NARCS)=BASELC(I2120)
     IF (OPDECL(I2120).EQ.LOCAT(1)) CAP(NARCS)=DEPOLC(I2120)
2120 CONTINUE
C
C      CREATE ARCS EMANATING FROM THE NRLA SRU NODES
     IF (SFMS.LT.1) GOTO 2140
     DO 2130 I2130=1,SFMS
C      CREATE AN ARC FROM THE NRLA SRU TO NRLA LRU IF THE
C      NRLA DECISION IS EITHER LRU-D, SRU-D OR LRU-B, SRU-B
     IF (OPDECL(LRUPTR(I2130)).EQ.LOCAT(1).AND.OPDECS(I2130).EQ.
```

```
     &LOCAT(1)) GOTO 2132
      IF (OPDECL(LRUPTR(I2130)).EQ.LOCAT(3).AND.OPDECS(I2130).EQ.
     &LOCAT(3)) GOTO 2132
      GOTO 2135
 2132 NARCS=NARCS+1
      SRCE(NARCS)=LNLRU+I2130
      DEST(NARCS)=LCSRU+LRUPTR(I2130)
      CAP(NARCS)=JUMBO
      IF (OPDECS(I2130).EQ.LOCAT(3)) GOTO 2135
      CAP(NARCS)=BRLSS(I2130)+BRLDS(I2130)
C
C      CREATE AN ARC FROM THE NRLA SRU NODE TO THE LAST NODE (SINK) --
C      ITEM RELATED COST OF NRLA REPAIR FOR THE SRU
 2135 NARCS=NARCS+1
      SRCE(NARCS)=LNLRU+I2130
      DEST(NARCS)=LNODE
      CAP(NARCS)=JUMBO
      IF(OPDECS(I2130).EQ.LOCAT(3)) CAP(NARCS)=BASESC(I2130)
      IF (OPDECS(I2130).EQ.LOCAT(1)) CAP(NARCS)=DEPOSC(I2130)
      IF (OPDECL(LRUPTR(I2130)).NE.LOCAT(1).AND.OPDECS(I2130).EQ.
     &LOCAT(1)) CAP(NARCS)=CAP(NARCS)+BRLSS(I2130)+BRLDS(I2130)
 2130 CONTINUE
C
C      CREATE ARCS TO AND FROM THE NRLA SE NODES
C      ADD ARCS FOR THE NRLA DEPOT SE DECISIONS
 2140 IF (NDSE.LT.1) GOTO 2160
C      FOR EACH DEPOT SE --
      DO 2150 I2150=1,NDSE
      IPTR=SEPF(I2150)
      IF (IPTR.EQ.0) GOTO 2149
 2145 ITEM=SEXREF(IPTR)
      IF (ITEM.LT.0.AND.OPDECS(IABS(ITEM)).NE.LOCAT(1)) GOTO 2148
      IF (ITEM.GT.0.AND.OPDECL(ITEM).NE.LOCAT(1)) GOTO 2148
      IF (ITEM.LT.0) ITEM=LFMS-ITEM
      NARCS=NARCS+1
      SRCE(NARCS)=LCSRU+ITEM
      DEST(NARCS)=LNSRU+I2150
      CAP(NARCS)=JUMBO
 2148 NPTR=NXTITM(IPTR)
      IF (NPTR.EQ.0) GOTO 2149
      IPTR=NPTR
      GOTO 2145
C      ARCS FROM DEPOT SE TO SINK
 2149 NARCS=NARCS+1
      SRCE(NARCS)=LNSRU+I2150
      DEST(NARCS)=LNODE
      CAP(NARCS)=SECOST(I2150)
 2150 CONTINUE
C      ADD ARCS FOR NRLA BASE DECISIONS
 2160 IF (NBSE.LT.1) GOTO 2180
C      FOR EACH BASE SE
      NDSEP1=NDSE+1
```

```
      DO 2170 I2170=NBSEP1,NBSEP0
      IPTR=SEPF(I2170)
      IF (IPTR.EQ.0) GOTO 2169
 2165 ITEM=SEXREF(IPTR)
      IF (ITEM.LT.0.AND.OPDECS(IABS(ITEM)).NE.LOCAT(3)) GOTO 2168
      IF (ITEM.GT.0.AND.OPDECL(ITEM).NE.LOCAT(3)) GOTO 2168
      IF (ITEM.LT.0) ITEM=LEMS-ITEM
      NARCS=NARCS+1
      SRCE(NARCS)=LCSRU+ITEM
      DEST(NARCS)=LNSRU+I2170
      CAP(NARCS)=JUMBO
 2168 NPTR=NXTITM(IPTR)
      IF (NPTR.EQ.0) GOTO 2169
      IPTR=NPTR
      GOTO 2165
C      ARCS FROM BASE SE TO SINK
 2169 NARCS=NARCS+1
      SRCE(NARCS)=LNSRU+I2170
      DEST(NARCS)=LNODE
      CAP(NARCS)=SECOST(I2170)
 2170 CONTINUE
 2180 NNODES=LNODE
C
C      EXPLICITLY SORT THE SRCE-DEST-CAP TRIPLES INTO ASCENDING ORDER
C      BY SRCE  (FLOW IS USED BY SORT TO STORE ORDERING POINTERS)
      CALL SORT(SRCE,DEST,CAP,FLOW,NARCS,MAXARC,1)
C      SAVE POINTERS FOR FORWARD SCAN IN FWDSP
      DO 2190 I2190=1,NARCS
C      UTILIZE THE VALUES IN SRCE IN REVERSE ORDER.  (IF NODE '1' IS A
C      SOURCE FOR 'N' ARCS THEN FWDSP(1) WILL SUCCESSIVELY GET THE
C      VALUES: N, N-1, N-2, . . . , 2, 1.  SIMILARLY, FWDSP(K) WILL
C      HAVE THE VALUE 'J' WHERE J CORRESPONDS TO THE FIRST OCCURENCE
C      OF NODE # K IN SRCE.)
      J2190=NARCS+1-I2190
      K2190=SRCE(J2190)
      FWDSP(K2190)=J2190
 2190 CONTINUE
      FWDSP(LNODE)=NARCS+1
      LNODM1=LNODE-1
      DO 2195 I21[illegible],LNODM1
      J2195=LNO[illegible]
      IF(FWDSP(J2[illegible]0)  FWDSP(J2195)=FWDSP(J2195+1)
 2195 CONTINUE
C
C      USE SORT T[illegible]ITLY SEQUENCE THE DEST ENTRIES INTO ASCENDING
C      ORDER.  THE ORDER WILL BE SPECIFIED BY THE ENTRIES IN BKPTR.
C      THUS, THE ENTRIES IN BACKSP ARE POINTERS TO SRCE-DEST-CAP
C      TRIPLES BASED ON THE DEST VALUES.
      CALL SORT(DEST,0,0,BKPTR,NARCS,MAXARC,0)
C      SAVE POINTERS FOR BACKWARD SCAN IN BACKSP
      BACKSP(1)=0
      DO 2200 I2200=1,NARCS
```

```
      J2200=BKPTR(I2200)
      K2200=DEST(J2200)
C      BACKSP(K) IS THE POSITION IN BKPTR WHICH HOLDS THE LAST POINTER
C      TO A SRCE-DEST-CAP TRIPLE HAVING NODE # K AS THE DEST.  THE FIRST
C      POINTER TO A TRIPLE HAVING NODE # K AS THE DEST IS BKPTR(K-1)+1.
      BACKSP(K2200)=I2200
 2200 CONTINUE
      DO 2222 I2222=2,LNODE
      IF(BACKSP(I2222).EQ.0)  BACKSP(I2222)=BACKSP(I2222-1)
 2222 CONTINUE
C
C IF(MSGL2.LT.1)  GO TO 2205
C PRINT 2202,'SRCE  ',(SRCE(I),I=1,NARCS)
C 2202 FORMAT(' ',A6,(' ',20I3))
C PRINT 2202,'DEST  ',(DEST(I),I=1,NARCS)
C PRINT 2202,'CAP   ',(CAP(I),I=1,NARCS)
C PRINT 2202,'FLOW  ',(FLOW(I),I=1,NARCS)
C PRINT 2202,'BKPTR ',(BKPTR(I),I=1,NARCS)
C PRINT 2202,'FWDSP ',(FWDSP(I),I=1,LNODE)
C PRINT 2202,'BACKSP',(BACKSP(I),I=1,LNODE)
 2205 DO 2400 I2400=1,NARCS
      J=SRCE(I2400)
      K=DEST(I2400)
      IF(J.NE.1)  GO TO 2230
      IF(K.GT.LCSE)  GO TO 2210
      SEARCP(K-1)=I2400
      GO TO 2300
 2210 IF(K.GT.LCLRU)  GO TO 2220
      LDARC(K-LCSE)=I2400
      GO TO 2300
 2220 SDARC(K-LCLRU)=I2400
      GO TO 2300
 2230 IF(J.LE.LCSE)  GO TO 2300
      IF(J.GT.LCLRU)  GO TO 2250
      IF(K.LE.LCSRU)  GO TO 2240
      LSARC(J-LCSE)=I2400
      GO TO 2300
 2240 SBDARC(K-LCLRU)=I2400
      GO TO 2300
 2250 IF(J.GT.LCSRU)  GO TO 2260
      SSARC(J-LCLRU)=I2400
      GO TO 2300
 2260 IF(J.GT.LNLRU)  GO TO 2280
      IF(K.NE.LNODE)  GO TO 2270
      LBARC(J-LCSRU)=I2400
      GO TO 2300
 2270 IF(K.GT.LNSRU)  GO TO 2300
      SBSARC(K-LNLRU)=I2400
      GO TO 2300
 2280 IF(J.GT.LNSRU)  GO TO 2290
      IF(K.NE.LNODE)  GO TO 2300
```

```
      SBARC(J-LNLRU)=I2400
      GO TO 2300
 2290 SEARCP(J+NCSE-LNSRU)=I2400
 2300 FLOW(I2400)=0
 2400 CONTINUE
C IF(MSGL2.LT.2)  GO TO 2440
C DO 2410 I=1,LFMS
C PRINT,'LPTRS ',LDARC(I),LSARC(I),LBARC(I)
C 2410 CONTINUE
C DO 2420 I=1,SFMS
C PRINT,'SPTRS ',SDARC(I),SSARC(I),SBARC(I),SBDARC(I),SBSARC(I)
C 2420 CONTINUE
C DO 2430 I=1,NUMSER
C PRINT,'SEPTRS ',SEARCP(I)
C 2430 CONTINUE
C 2440 CONTINUE
      RETURN
      END
```

*

Appendix D: MARLA Subroutine

```
C
      SUBROUTINE MARLA
C
*     THIS PROGRAM IMPLEMENTS THE CONCEPT OF MARGINAL      *
*     ANALYSIS TO DETERMINE THE OPTIMUM REPAIR LEVEL       *
*     FOR AN ITEM (LRU FAILURE MODE OR SRU) OF A WEAPONS   *
*     SYSTEM. IT WILL ACT AS A PREPROCESSER TO THE MAXFLO  *
*     SUBROUTINE OF THE NRLA PROGRAM.                      *
C
*      DO LOOP 10 CONDUCTS THE ANALYSIS FOR THE FIRST ITEM  *
*      TO THE LAST ITEM OF THE SYSTEM.                      *
C
       TOTITM=LFMS+SFMS
       DO 10 I=1,TOTITM
       IF(I.GT.LFMS) GO TO 15
       IF(LFMOD(I).NE.1) GO TO 12
        LFMOD(I)=2
        INDD=1000
    12 IF(LFMOB(I).NE.1) GO TO 14
        LFMOB(I)=2
        INDB=10
    14 IF(LFMOC(I).NE.1) GO TO 16
        LFMOC(I)=2
        INDC=1
    16 IF(LFMOS(I).NE.1) GO TO 18
        LFMOS(I)=2
        INDS=100
    18 CONTINUE
       IF (CIRF.EQ.0) LFMOC(I)=3
C
*      DO LOOP 20 CHECKS EACH POSITION OF THE SE/LRU/SRU      *
*      CROSS REFERENCE ARRAY (SEARY) FOR THE POINTER VALUES  *
*      OF THE SE NECESSARY TO REPAIR THE Ith ITEM.           *
C
       DO 20 I20=1,26
         PTR=SEARY(I,I20)
         IF(PTR.EQ.0) GO TO 20
         ID=SECODE(PTR)/10
       IF(LFMOD(I).GT.0.OR.ID.GE.5000) GO TO 32
       BUY=AINT(1.0+(FSEUHD(I)/OPHRS(PTR)))
       TSECD(I)=TSECD(I)+(BUY*CADB(PTR)+FDB(PTR)+(FSEUHD(I)/
      & OPHRS(PTR))*CODB(PTR)*PIUP)
    32 IF(LFMOB(I).GT.0.OR.ID.LT.5000.OR.ID.GE.9000) GO TO 34
       BUY=AINT(1.0+(FSEUHB(I)/OPHRS(PTR)))
       TSECB(I)=TSECB(I)+(BUY*CADB(PTR)+FDB(PTR)+((FSEUHB(I)/
      & OPHRS(PTR))*CODB(PTR)*PIUP))
    34 IF(LFMOC(I).GT.0.OR.ID.LT.9000) GO TO 20
       BUY=2+AINT((FSEUHC(I)*OS)/OPHRS(PTR))+AINT((FSEUHC(I)*(1-OS))
      & /OPHRS(PTR))
       TSECC(I)=TSECC(I)+(BUY*CADB(PTR)+FDB(PTR)+(FSEUHC(I)/
```

```
     & OPHRS(PTR))*COOB(PTR)*PI(P)
  20 CONTINUE
C
C     NOW THE TOTAL COST FOR EACH ITEM                              *
*     DETERMINED (TDCST,TBCST,AND TCCST). FINALLY USING A           *
*     SERIES OF IF STATEMENTS A MARGINAL ANALYSIS IS                *
*     PERFORMED TO IDENTIFY IF ONE REPAIR LEVEL IS MORE             *
*     ECONOMICAL THAN ANOTHER. IF THIS IS TRUE A UNIQUE             *
*     INDICATER IS SET (INDD,INDB,INDS,INDC).                       *
C
      IF(TSECD(I).EQ.0) GO TO 38
      SED=TSECD(I)/M
      IF(TSECC(I).EQ.0) GO TO 38
      SEC=TSECC(I)/M
  38 TDCST(I)=DEPOLC(I)+SED
      TBCST(I)=BASELC(I)+TSECB(I)
      IF(CIRF.EQ.0) TCCST(I)=JUMBO
      TCCST(I)=CIRFLC(I)+SEC
        IF(LEMOD(I).GT.0.OR.LEMOS(I).GT.0) GO TO 80
        IF(SCRPLC(I).LT.DEPOLC(I))  INDD=1000
        IF(TDCST(I).LT.SCRPLC(I)) INDS=100
  80 IF(LEMOS(I).GT.0.OR.LEMOB(I).GT.0) GO TO 82
        IF(TBCST(I).LT.SCRPLC(I)) INDS=100
        IF(SCRPLC(I).LT.BASELC(I)) INDB=10
  82 IF(LEMOB(I).GT.0.OR.LEMOD(I).GT.0) GO TO 84
        IF(TBCST(I).LT.DEPOLC(I)) INDD=1000
        IF(TDCST(I).LT.BASELC(I)) INDB=10
  84 IF(LEMOC(I).GT.0.OR.LEMOD(I).GT.0) GO TO 74
        IF(TCCST(I).LT.DEPOLC(I)) INDD=1000
        IF(TDCST(I).LT.CIRFLC(I)) INDC=1
  74 IF(LEMOC(I).GT.0.OR.LEMOS(I).GT.0) GO TO 77
        IF(SCRPLC(I).LT.CIRFLC(I)) INDC=1
        IF(TCCST(I).LT.SCRPLC(I)) INDS=100
  77 IF(LEMOC(I).GT.0.OR.LEMOB(I).GT.0) GO TO 87
        IF(TCCST(I).LT.BASELC(I)) INDB=10
        IF(TBCST(I).LT.CIRFLC(I)) INDC=1
C
C     LINES BELOW CHECK THE VALUES OF THE INDICATERS                *
*     IF THE INDICATER HAS BEEN GIVEN THE RIGHT VALUE, THE *
*     NRLA FAILURE MODE EXCLUSION ARRAY IS UPDATED TO               *
*     ELIMINATE THAT REPAIR OPTION FROM FURTHER CONSIDER-  *
*     ATION FOR THE Ith ITEM.                                       *
C
  87 IF(LEMOD(I).EQ.2) GO TO 81
     IF(INDD.EQ.1000) LEMOD(I)=1
  81 IF(LEMOS(I).EQ.2) GO TO 83
     IF(INDS.EQ.100 ) LEMOS(I)=1
  83 IF(LEMOB(I).EQ.2) GO TO 85
     IF(INDB.EQ.10)    LEMOB(I)=1
   85 IF(LEMOC(I).GE.2) GO TO 88
      IF(INDC.EQ.1)     LEMOC(I)=1
   88 TYPRPR=INDD+INDS+INDB+INDC
C
```

```
*     BASED ON THE VALUE OF TYPRPR THE BELOW STATEMENTS       *
*     CHECK TO SEE IF 3 OUT OF THE 4 POSSIBLE OPTIONS HAVE *
*     BEEN ELIMINATED. IF THIS IS TRUE THE OPTIMAL DECISION*
*     HAS BEEN MADE AND IS STORED IN THE ARRAY 'OPDECL'      *
C
       IF(TYPRPR.EQ.0111) OPDECL(I)=LOCAT(1)
       IF(TYPRPR.EQ.1011) OPDECL(I)=LOCAT(2)
       IF(TYPRPR.EQ.1101) OPDECL(I)=LOCAT(3)
       IF(TYPRPR.EQ.1110) OPDECL(I)=LOCAT(4)
      TSECD(I)=0
      TSECB(I)=0
      TSECC(I)=0
C
*     DO LOOP 25 UPDATES THE NUMBER OF PIECES OF SE AT EACH *
*     LEVEL BASED ON THE REPAIR LEVEL DECISIONS THAT HAVE     *
*     BEEN MADE AS WELL AS THE SE HOURS STILL AVAILABLE.      *
*     IT THEN GOES BACK AND CHECKS TO MAKE SURE THAT ONLY     *
*     ONE ADDITIONAL PIECE OF SE IS SUFFICIENT.               *
C
      DO 25 I25=1,26
       PTR=SEARY(I,I25)
       IF(PTR.EQ.0) GO TO 25
      HRAVSE(PTR)=NSECI(PTR)*(OPHRS(PTR)-BSYHRS(PTR))
       ID=SECODE(PTR)/10
      IF(LFMOD(I).GT.0) GO TO 41
       IF(TYPRPR.EQ.0111.AND.ID.LT.5000) GO TO 42
   41 IF(LFMOB(I).GT.0) GO TO 48
       IF(TYPRPR.EQ.1101.AND.ID.GE.5000.AND.ID.LT.9000) GO TO 44
   48 IF(LFMOC(I).GT.0) GO TO 10
       IF(TYPRPR.EQ.1110.AND.ID.GE.9000) GO TO 46
      GO TO 25
   42 IF(HRAVSE(PTR).GE.FSEUHD(I)) GO TO 43
       REQMT(PTR)=REQMT(PTR)+1
       HRAVSE(PTR)=HRAVSE(PTR)+OPHRS(PTR)
       GO TO 42
   43 USEHRS(PTR)=USEHRS(PTR)+FSEUHD(I)
      FAC=FDB(PTR)
      IF(NSECI(PTR).GT.0) FAC=0.
      TSECST(PTR)=TSECST(PTR)+((REQMT(PTR)*CADB(PTR)+FAC+
     & ((FSEUHD(I)/OPHRS(PTR))*CODB(PTR)*PIUP))/FLOAT(M))
      HRAVSE(PTR)=HRAVSE(PTR)-FSEUHD(I)
      GO TO 25
   44 IF(HRAVSE(PTR).GE.FSEUHB(I)) GO TO 45
       REQMT(PTR)=REQMT(PTR)+1
       HRAVSE(PTR)=HRAVSE(PTR)+OPHRS(PTR)
       GO TO 44
   45 USEHRS(PTR)=USEHRS(PTR)+FSEUHB(I)
      FAC=FDB(PTR)
      IF(NSECI(PTR).GT.0) FAC=0.
      TSECST(PTR)=TSECST(PTR)+(REQMT(PTR)*CADB(PTR)+FAC+((FSEUHB(I)
     & /OPHRS(PTR))*CODB(PTR)*PIUP))
      HRAVSE(PTR)=HRAVSE(PTR)-FSEUHB(I)
      GO TO 25
```

```
   46 IF(HRAVSE(PTR).GE.FSEUHC(I)) GO TO 47
       REQMT(PTR)=2+AINT((FSEUHC(I)*OS)/OPHRS(PTR))+
     & AINT((FSEUHC(I)*(1-OS))/OPHRS(PTR))
       HRAVSE(PTR)=HRAVSE(PTR)+(REQMT(PTR)*OPHRS(PTR))
   47 USEHRS(PTR)=USEHRS(PTR)+FSEUHC(I)
      FAC=FDB(PTR)
      IF(NSECI(PTR).GT.0) FAC=0.
      TSECST(PTR)=TSECST(PTR)+((REQMT(PTR)*CADB(PTR)+FAC+
     & ((FSEUHC(I)/OPHRS(PTR))*CODB(PTR)*PIUP))/FLOAT(M))
      HRAVSE(PTR)=HRAVSE(PTR)-FSEUHC(I)
   25 CONTINUE
C
*     IF THE CURRENT ITEM IS AN LRU FM GO TO 10                  *
*     BECAUSE THE FOLLOWING STATEMENTS RELATE ONLY TO SRU'S.     *
C
      GO TO 10
   15  CONTINUE
      J=I-LFMS
      IF(OPDECL(LRUPTR(J)).EQ.LOCAT(15)) GO TO 10
      IF(SOD(J).NE.1) GO TO 22
       SOD(J)=2
       INDD=1000
   22 IF(SOB(J).NE.1) GO TO 24
       SOB(J)=2
       INDB=10
   24 IF(SOC(J).NE.1) GO TO 26
       SOC(J)=2
       INDC=1
   26 IF(SOS(J).NE.1) GO TO 28
       SOS(J)=2
       INDS=100
   28 CONTINUE
      IF(CIRF.EQ.0) SOC(I)=3
C
*     DO LOOP 20 CHECKS EACH POSITION OF THE SE/LRU/SRU          *
*     CROSS REFERENCE ARRAY (SEARY) FOR THE POINTER VALUES       *
*     OF THE SE NECESSARY TO REPAIR THE Ith ITEM.                *
C
      DO 50 I50=1,26
        PTR=SEARY(I,I50)
        IF(PTR.EQ.0) GO TO 50
       ID=SECODE(PTR)/10
      IF(SOD(J).GT.0.OR.ID.GE.5000) GO TO 52
      BUY=AINT(1.0+(SSEUHD(J)/OPHRS(PTR)))
      TSECD(J)=TSECD(J)+(BUY*CADB(PTR)+FDB(PTR)+(SSEUHD(J)/
     & OPHRS(PTR))*CODB(PTR)*PIUP)
   52 IF(SOB(J).GT.0.OR.ID.LT.5000.OR.ID.GE.9000) GO TO 54
      BUY=AINT(1.0+(SSEUHB(J)/OPHRS(PTR)))
      TSECB(J)=TSECB(J)+(BUY*CADB(PTR)+FDB(PTR)+((SSEUHB(J)/
     & OPHRS(PTR))*CODB(PTR)*PIUP))
   54 IF(SOC(J).GT.0.OR.ID.LT.9000) GO TO 50
      BUY=2+AINT((SSEUHC(J)*OS)/OPHRS(PTR))+AINT((SSEUHC(J)*(1-OS))
     & /OPHRS(PTR))
```

```
C     TSECC(J)=TSECC(J)+(BUY*CADB(PTR)+FDB(PTR)+([illegible](J)/
     & OPHRS(PTR))*CODB(PTR)*PIU2)
   50 CONTINUE
C
C          NOW THE TOTAL COST FOR EACH ITEM IS
*     DETERMINED (TDCST,TBCST,AND TCCST). FINALLY USING A    *
*     SERIES OF IF STATEMENTS A MARGINAL ANALYSIS IS         *
*     PERFORMED TO IDENTIFY IF ONE REPAIR LEVEL IS MORE      *
*     ECONOMICAL THAN ANOTHER. IF THIS IS TRUE A UNIQUE      *
*     INDICATER IS SET (INDD,INDB,INDS,INDC).                *
C
      IF(TSECD(J).EQ.0) GO TO 58
      SED=TSECD(J)/M
      IF(TSECC(J).EQ.0) GO TO 58
      SEC=TSECC(J)/M
   58 TDCST(J)=DEPOSC(J)+SED
      IF(OPDECL(LRUPTR(J)).EQ.LOCAT(3)) TDCST(J)=TDCST(J)+
     & BRLSS(J)+BRLDS(J)
      TBCST(J)=BASESC(J)+TSECB(J)
      TCCST(J)=CIRFSC(J)+SEC
      IF(OPDECL(LRUPTR(J)).EQ.LOCAT(4)) TCCST(J)=TCCST(J)+BRLCS(J)
      IF(OPDECL(LRUPTR(J)).EQ.LOCAT(3)) SCRPSC(J)=SCRPSC(J)+BRLSS(J)
      IF(CIRF.EQ.0) TCCST(J)=JUMBO
        IF(SOD(J).GT.0.OR.SOS(J).GT.0) GO TO 60
        IF(SCRPSC(J).LT.DEPOSC(J))  INDD=1000
        IF(TDCST(J).LT.SCRPSC(J)) INDS=100
   60 IF(SOS(J).GT.0.OR.SOB(J).GT.0) GO TO 62
        IF(TBCST(J).LT.SCRPSC(J)) INDS=100
        IF(SCRPSC(J).LT.BASESC(J)) INDB=10
   62 IF(SOB(J).GT.0.OR.SOD(J).GT.0) GO TO 64
        IF(TBCST(J).LT.DEPOSC(J)) INDD=1000
        IF(TDCST(J).LT.BASESC(J)) INDB=10
   64 IF(SOC(J).GT.0.OR.SOD(J).GT.0) GO TO 67
        IF(TCCST(J).LT.DEPOSC(J)) INDD=1000
        IF(TDCST(J).LT.CIRFSC(J)) INDC=1
   67 IF(SOC(J).GT.0.OR.SOS(J).GT.0) GO TO 68
        IF(SCRPLC(J).LT.CIRFSC(J)) INDC=1
        IF(TCCST(J).LT.SCRPSC(J)) INDS=100
   68 IF(SOC(J).GT.0.OR.SOB(J).GT.0) GO TO 69
        IF(TCCST(J).LT.BASESC(J)) INDB=10
        IF(TBCST(J).LT.CIRFSC(J)) INDC=1
      IF(OPDECL(LRUPTR(J)).EQ.LOCAT(3)) SCRPSC(J)=SCRPSC(J)
     & -BRLSS(J)
C
C     LINES BELOW CHECK THE VALUES OF THE INDICATERS         *
*     IF THE INDICATER HAS BEEN GIVEN THE RIGHT VALUE, THE   *
*     NRLA FAILURE MODE EXCLUSION ARRAY IS UPDATED TO        *
*     ELIMINATE THAT REPAIR OPTION FROM FURTHER CONSIDER-    *
*     ATION FOR THE Ith ITEM.                                *
C
   69 IF(SOD(J).EQ.2) GO TO 61
      IF(INDD.EQ.1000) SOD(J)=1
   61 IF(SOS(J).EQ.2) GO TO 63
```

```
      IF(INDS.EQ.100 ) SOS(J)=1
   63 IF(SOB(J).EQ.2) GO TO 65
      IF(INDB.EQ.10)   SOB(J)=1
   65 IF(SOC(J).GE.2) GO TO 66
      IF(INDC.EQ.1)    SOC(J)=1
   66 TYPRPR=INDD+INDS+INDB+INDC
C
*     BASED ON THE VALUE OF TYPRPR THE BELOW STATEMENTS     *
*     CHECK TO SEE IF 3 OUT OF THE 4 POSSIBLE OPTIONS HAVE  *
*     BEEN ELIMINATED. IF THIS IS TRUE THE OPTIMAL DECISION*
*     HAS BEEN MADE AND IS STORED IN THE ARRAY 'OPDECS'     *
C
       IF(TYPRPR.EQ.0111) OPDECS(J)=LOCAT(1)
       IF(TYPRPR.EQ.1011) OPDECS(J)=LOCAT(2)
       IF(TYPRPR.EQ.1101) OPDECS(J)=LOCAT(3)
       IF(TYPRPR.EQ.1110) OPDECS(J)=LOCAT(4)
      TSECD(J)=0
      TSECB(J)=0
      TSECC(J)=0
C
*     DO LOOP 75 UPDATES THE NUMBER OF PIECES OF SE AT EACH *
*     LEVEL BASED ON THE REPAIR LEVEL DECISIONS THAT HAVE   *
*     BEEN MADE AS WELL AS THE SE HOURS STILL AVAILABLE.    *
*     IT THEN GOES BACK AND CHECKS TO MAKE SURE THAT ONLY   *
*     ONE ADDITIONAL PIECE OF SE IS SUFFICIENT.             *
C
      DO 75 I75=1,26
       PTR=SEARY(I,I75)
       IF(PTR.EQ.0) GO TO 75
      HRAVSE(PTR)=NSECI(PTR)*(OPHRS(PTR)-BSYHRS(PTR))
       ID=SECODE(PTR)/10
      IF(SOD(J).GT.0) GO TO 91
       IF(TYPRPR.EQ.0111.AND.ID.LT.5000) GO TO 92
   91 IF(SOB(J).GT.0) GO TO 98
       IF(TYPRPR.EQ.1101.AND.ID.GE.5000.AND.ID.LT.9000) GO TO 94
   98 IF(SOC(J).GT.0) GO TO 10
       IF(TYPRPR.EQ.1110.AND.ID.GE.9000) GO TO 96
      GO TO 75
   92 IF(HRAVSE(PTR).GE.SSEUHD(J)) GO TO 93
       REQMT(PTR)=REQMT(PTR)+1
       HRAVSE(PTR)=HRAVSE(PTR)+OPHRS(PTR)
       GO TO 92
   93 USEHRS(PTR)=USEHRS(PTR)+SSEUHD(J)
      FAC=FDB(PTR)
      IF(NSECI(PTR).GT.0) FAC=0.
      TSECST(PTR)=TSECST(PTR)+((REQMT(PTR)*CADB(PTR)+FAC+
     & ((SSEUHD(J)/OPHRS(PTR))*CODB(PTR)*PIUP))/FLOAT(M))
      HRAVSE(PTR)=HRAVSE(PTR)-SSEUHD(J)
      GO TO 75
   94 IF(HRAVSE(PTR).GE.SSEUHB(J)) GO TO 95
       REQMT(PTR)=REQMT(PTR)+1
       HRAVSE(PTR)=HRAVSE(PTR)+OPHRS(PTR)
       GO TO 94
   95 USEHRS(PTR)=USEHRS(PTR)+SSEUHB(J)
      FAC=FDB(PTR)
```

```
      IF(NSECI(PTR).GT.0) FAC=0.
      TSECST(PTR)=TSECST(PTR)+(REQMT(PTR)*CAD3(PTR)+FAC+((SSEUHB(J)
     & /OPHRS(PTR))*CODB(PTR)*PIUP))
      HRAVSE(PTR)=HRAVSE(PTR)-SSEUHB(J)
      GO TO 75
   96 IF(HRAVSE(PTR).GE.SSEUHC(J)) GO TO 97
       REQMT(PTR)=2+AINT((SSEUHC(J)*OS)/OPHRS(PTR))+
     & AINT((SSEUHC(J)*(1-OS))/OPHRS(PTR))
       HRAVSE(PTR)=HRAVSE(PTR)+(REQMT(PTR)*OPHRS(PTR))
   97 USEHRS(PTR)=USEHRS(PTR)+SSEUHC(J)
      FAC=FDB(PTR)
      IF(NSECI(PTR).GT.0) FAC=0.
      TSECST(PTR)=TSECST(PTR)+((REQMT(PTR)*CAD3(PTR)+FAC+
     & ((SSEUHC(J)/OPHRS(PTR))*CODF(PTR)*PIUP))/FLOAT(M))
      HRAVSE(PTR)=HRAVSE(PTR)-SSEUHC(J)
   75 CONTINUE
      INDD=0
      INDC=0
      INDB=0
      INDS=0
   10 CONTINUE
      RETURN
      END
```

*

Appendix E: MSETNT Subroutine

```
      SUBROUTINE MSETNT(MSGL2)
C
C
C     CODE FROM STATEMENT # 2000 THROUGH # 2210 UTILIZES THE ABOVE
C     COMPUTED ITEM & SE COSTS TO PRODUCE AN RLA COST NETWORK.
C
C     AS A PRELIMINARY STEP DETERMINE NODE NUMBERS FOR KEY NODES IN
C     THE NETWORK.  THE VARIABLES ARE CODED WITH THE PREFIXES 'LD'
C     FOR 'LAST DEPOT' AND 'LB' FOR 'LAST BASE'.
C
 2000 LDSE=NDSE+1
      LDLRU=LDSE+LFMS
      LDSRU=LDLRU+SFMS
      LBLRU=LDSRU+LFMS
      LBSRU=LBLRU+SFMS
      LBSE=LBSRU+NBSE
      LNODE=LBSE+1
      IF(LNODE.LE.MAXNOD)  GO TO 2005
      WRITE(6,2002)
 2002 FORMAT('0 NODE VECTORS ARE TOO SMALL -- STOP')
      STOP
C
C
C      SET # OF ARCS TO ZERO & # OF NODES TO 1
 2005 NARCS=0
      NNODES=1
C
      IF(NDSE.LT.1)  GO TO 2015
C      CREATE ARCS FROM SOURCE TO DEPOT SE -- DEPOT SE COST ARCS
      DO 2010 I2010=1,NDSE
      NARCS=NARCS+1
      NNODES=NNODES+1
      SRCE(NARCS)=1
      DEST(NARCS)=NNODES
      CAP(NARCS)=SECOST(I2010)
 2010 CONTINUE
C
C      CREATE ARCS FROM THE SOURCE TO THE DEPOT LRU FAILURE MODE
C      NODES -- ITEM RELATED COSTS FOR LRU REPAIR AT DEPOT
 2015 DO 2020 I2020=1,LFMS
      NARCS=NARCS+1
      NNODES=NNODES+1
      SRCE(NARCS)=1
      DEST(NARCS)=NNODES
      CAP(NARCS)=DEPOLC(I2020)
 2020 CONTINUE
C
C      CREATE ARCS FROM THE SOURCE TO THE DEPOT SRU NODES -- ITEM
C      RELATED COSTS FOR SRU REPAIR AT DEPOT
      IF(SFMS.LT.1)  GO TO 2040
```

```
      DO 2030 I2030=1,SFMS
      NARCS=NARCS+1
      NNODES=NNODES+1
      SRCE(NARCS)=1
      DEST(NARCS)=NNODES
      CAP(NARCS)=DEPOSC(I2030)
 2030 CONTINUE
C
C      CREATE ARCS FROM THE DEPOT SE NODES TO THE DEPOT LRU/SRU NODES
 2040 IF(NDSE.LT.1)  GO TO 2070
C      FOR EACH DEPOT SE --
      DO 2060 I2060=1,NDSE
C      SEPF CONTAINS A POINTER TO THE SE CROSS REFERENCE (SEXREF)
      IPTR=SEPF(I2060)
      IF(IPTR.EQ.0)  GO TO 2060
C      SEXREF CONTAINS THE # OF AN LRU FAILURE MODE (ITEM > 0) OR
C      THE NUMBER OF AN SRU (ITEM < 0)
 2050 ITEM=SEXREF(IPTR)
      IF(ITEM.LT.0)  ITEM=LFMS-ITEM
      NARCS=NARCS+1
      SRCE(NARCS)=I2060+1
C      ITEM IS A POINTER TO AN LRU OR SRU AND WHEN ADDED TO LDSE
C      GIVES THE APPROPRIATE NODE NUMBER
      DEST(NARCS)=LDSE+ITEM
      CAP(NARCS)=JUMBO
C
C      NXTITM IS A POINTER TO THE NEXT ENTRY IN SEXREF WHICH IS AN
C      LRU OR SRU REQUIRING THE CURRENT SE, I.E., SE # I2060.
      NPTR=NXTITM(IPTR)
      IF(NPTR.EQ.0)  GO TO 2060
      IPTR=NPTR
      GO TO 2050
 2060 CONTINUE
C
C      CREATE THE ARCS EMANATING FROM THE DEPOT LRU FAILURE MODE NODES
 2070 DO 2080 I2080=1,LFMS
C      CHECK FOR AN SRU
      IPTR=SRUPTR(I2080)
      IF(IPTR.EQ.0)  GO TO 2075
C      CREATE AN ARC TO THE DEPOT SRU NODE -- COSTS UNIQUE TO BASE
C      REPAIR OF LRU & DEPOT REPAIR OF SRU
      NARCS=NARCS+1
      SRCE(NARCS)=LDSE+I2080
      DEST(NARCS)=LDLRU+IPTR
      CAP(NARCS)=BRLDS(IPTR)
C      CREATE AN ARC TO THE BASE LRU NODES -- COST OF SCRAPPING THE LRU
 2075 NARCS=NARCS+1
      SRCE(NARCS)=LDSE+I2080
      DEST(NARCS)=LDSRU+I2080
       CAP(NARCS)=SCRPLC(I2080)
 2080 CONTINUE
C
```

```
C       CREATE ARCS FROM THE DEPOT SRU NODES TO THE BASE SRU NODES --
C       COST OF SCRAPPING THE SRU
      IF(SFMS.LT.1)  GO TO 2100
      DO 2090 I2090=1,SFMS
      NARCS=NARCS+1
      SRCE(NARCS)=LDLRU+I2090
      DEST(NARCS)=LBLRU+I2090
      CAP(NARCS)=SCRPSC(I2090)
 2090 CONTINUE
C
C       CREATE ARCS EMANATING FROM THE BASE LRU NODES
 2100 DO 2120 I2120=1,LFMS
C       CHECK FOR AN SRU
      IPTR=SRUPTR(I2120)
      IF(IPTR.EQ.0)  GO TO 2110
C
C       CREATE THE ARC FROM THEBASE LRU NODE TO THE BASE SRU NODE --
C       COSTS UNIQUE TO BASE REPAIR OF THE LRU AND SCRAPPING (OR DEPOT
C       REPAIR) OF THE SRU
      NARCS=NARCS+1
      SRCE(NARCS)=LDSRU+I2120
      DEST(NARCS)=LBLRU+IPTR
      CAP(NARCS)=BRLSS(IPTR)
C       CREATE AN ARC FROM THE BASE LRU NODE TO THE LAST NODE (SINK) --
C       ITEM RELATED COST OF BASE REPAIR OF THE LRU
 2110 NARCS=NARCS+1
      SRCE(NARCS)=LDSRU+I2120
      DEST(NARCS)=LNODE
      CAP(NARCS)=BASELC(I2120)
 2120 CONTINUE
C
C       CREATE ARCS EMANATING FROM THE BASE SRU NODES
      IF(SFMS.LT.1)  GO TO 2140
      DO 2130 I2130=1,SFMS
C       CREATE AN ARC FROM THE BASE SRU NODE TO THE BASE LRU NODE --
C       THIS ARC PREVENTS A DECISION TO DO BASE LEVEL SRU REPAIR UNLESS
C       THE LRU IS ALSO BASE REPAIRED
      NARCS=NARCS+1
      SRCE(NARCS)=LBLRU+I2130
      DEST(NARCS)=LDSRU+LRUPTR(I2130)
      CAP(NARCS)=JUMBO
C
C       CREATE AN ARC FROM THE BASE SRU NODE TO THE LAST NODE (SINK) --
C       ITEM RELATED COST OF BASE REPAIR FOR THE SRU
      NARCS=NARCS+1
      SRCE(NARCS)=LBLRU+I2130
      DEST(NARCS)=LNODE
      CAP(NARCS)=BASESC(I2130)
 2130 CONTINUE
C
C       CREATE ARCS TO AND FROM THE BASE SE NODES
 2140 IF(NBSE.LT.1)  GO TO 2180
C       FOR EACH BASE SE --
```

```
      NDSEP1=NDSE+1
      NBSEPO=NDSE+NBSE
      DO 2170 I2170=NDSEP1,NBSEPO
C      J2170 IS A POINTER TO THE BASE SE
      J2170=I2170-NDSE
C      SEPF CONTAINS A POINTER TO THE SE CROSS REFERENCE (SEXREF)
C        ( SEE COMMENTS FOR STATEMENTS 2040 TO 2060)
      IPTR=SEPF(I2170)
      IF(IPTR.EQ.0)  GO TO 2160
 2150 ITEM=SEXREF(IPTR)
      IF(ITEM.LT.0)  ITEM=LFMS-ITEM
      NARCS=NARCS+1
      SRCE(NARCS)=LDSRU+ITEM
      DEST(NARCS)=LBSRU+J2170
      CAP(NARCS)=JUMBO
C
      NPTR=NXTITM(IPTR)
      IF(NPTR.EQ.0)  GO TO 2160
      IPTR=NPTR
      GO TO 2150
C
C      CREATE AN ARC FROM THE BASE SE TO THE LAST NODE (SINK) --
C      BASE SE COST
 2160 NARCS=NARCS+1
      SRCE(NARCS)=LBSRU+J2170
      DEST(NARCS)=LNODE
      CAP(NARCS)=SECOST(I2170)
 2170 CONTINUE
 2180 NNODES=LNODE
C
C       EXPLICITLY SORT THE SRCE-DEST-CAP TRIPLES INTO ASCENDING ORDER
C       BY SRCE  (FLOW IS USED BY SORT TO STORE ORDERING POINTERS)
       CALL SORT(SRCE,DEST,CAP,FLOW,NARCS,MAXARC,1)
C      SAVE POINTERS FOR FORWARD SCAN IN FWDSP
      DO 2190 I2190=1,NARCS
C      UTILIZE THE VALUES IN SRCE IN REVERSE ORDER.  (IF NODE '1' IS A
C      SOURCE FOR 'N' ARCS THEN FWDSP(1) WILL SUCCESSIVELY GET THE
C      VALUES: N, N-1, N-2, . . . . , 2, 1.  SIMILARLY, FWDSP(K) WILL
C      HAVE THE VALUE 'J' WHERE J CORRESPONDS TO THE FIRST OCCURENCE
C      OF NODE # K IN SRCE.)
       J2190=NARCS+1-I2190
       K2190=SRCE(J2190)
       FWDSP(K2190)=J2190
 2190  CONTINUE
      FWDSP(LNODE)=NARCS+1
      LNODM1=LNODE-1
      DO 2195 I2195=1,LNODM1
      J2195=LNODE-I2195
      IF(FWDSP(J2195).EQ.0)  FWDSP(J2195)=FWDSP(J2195+1)
 2195 CONTINUF
C
C      USE SORT TO IMPLICITLY SEQUENCE THE DEST ENTRIES INTO ASCENDING
C      ORDER.  THE ORDER WILL BE SPECIFIED BY THE ENTRIES IN BKPTR.
```

```
C      THUS, THE ENTRIES IN BACKSP ARE POINTERS TO SRCE-DEST-CAP
C      TRIPLES BASED ON THE DEST VALUES.
      CALL SORT(DEST,0,0,BKPTR,NARCS,MAXARC,0)
      BACKSP(1)=0
      DO 2200 I2200=1,NARCS
      J2200=BKPTR(I2200)
      K2200=DEST(J2200)
C      BACKSP(K) IS THE POSITION IN BKPTR WHICH HOLDS THE LAST POINTER
C      TO A SRCE-DEST-CAP TRIPLE HAVING NODE # K AS THE DEST.  THE FIRST
C      POINTER TO A TRIPLE HAVING NODE # K AS THE DEST IS BKPTR(K-1)+1.
      BACKSP(K2200)=I2200
 2200 CONTINUE
      DO 2222 I2222=2,LNODE
      IF(BACKSP(I2222).EQ.0)  BACKSP(I2222)=BACKSP(I2222-1)
 2222 CONTINUE
C
C IF(MSGL2.LT.1)  GO TO 2205
C PRINT 2202,'SRCE  ',(SRCE(I),I=1,NARCS)
C 2202 FORMAT(' ',A6,(' ',20I3))
C PRINT 2202,'DEST  ',(DEST(I),I=1,NARCS)
C PRINT 2202,'CAP   ',(CAP(I),I=1,NARCS)
C PRINT 2202,'FLOW  ',(FLOW(I),I=1,NARCS)
C PRINT 2202,'BKPTR ',(BKPTR(I),I=1,NARCS)
C PRINT 2202,'FWDSP ',(FWDSP(I),I=1,LNODE)
C PRINT 2202,'BACKSP',(BACKSP(I),I=1,LNODE)
 2205 DO 2400 I2400=1,NARCS
      J=SRCE(I2400)
      K=DEST(I2400)
      IF(J.NE.1)  GO TO 2230
      IF(K.GT.LDSE)  GO TO 2210
      SEARCP(K-1)=I2400
      GO TO 2300
 2210 IF(K.GT.LDLRU)  GO TO 2220
      LDARC(K-LDSE)=I2400
      GO TO 2300
 2220 SDARC(K-LDLRU)=I2400
      GO TO 2300
 2230 IF(J.LE.LDSE)  GO TO 2300
      IF(J.GT.LDLRU)  GO TO 2250
      IF(K.LE.LDSRU)  GO TO 2240
      LSARC(J-LDSE)=I2400
      GO TO 2300
 2240 SBDARC(K-LDLRU)=I2400
      GO TO 2300
 2250 IF(J.GT.LDSRU)  GO TO 2260
      SSARC(J-LDLRU)=I2400
      GO TO 2300
 2260 IF(J.GT.LBLRU)  GO TO 2280
      IF(K.NE.LNODE)  GO TO 2270
      LBARC(J-LDSRU)=I2400
      GO TO 2300
 2270 IF(K.GT.LBSRU)  GO TO 2300
      SBSARC(K-LBLRU)=I2400
```

```
          GO TO 2300
     2280 IF(J.GT.LBSRU)  GO TO 2290
          IF(K.NE.LNODE)  GO TO 2300
          SBARC(J-LBLRU)=I2400
          GO TO 2300
     2290 SEARCP(J+NDSE-LBSRU)=I2400
     2300 FLOW(I2400)=0
     2400 CONTINUE
C
C          THIS ROUTINE USED ONLY IF MARLA IS CALLED IN MAIN
C
          IF(MAR.EQ.0) GO TO 3000
          DO 2450 I2450=1,MAXNOD
           FWDSP(I2450)=0
           BACKSP(I2450)=0
     2450 CONTINUE
          DO 2460 I2460=1,MAXARC
           BKPTR(I2460)=0
     2460 CONTINUE
C
           DO 2181 I2181=1,LFMS
          IF(OPDECL(I2181).EQ.LOCAT(15)) GO TO 2181
            IPTR=LDARC(I2181)
            SRCE(IPTR)=LNODE+10
            IPTR=LBARC(I2181)
            SRCE(IPTR)=LNODE+10
            IPTR=LSARC(I2181)
            SRCE(IPTR)=LNODE+10
          IF(OPDECL(I2181).NE.LOCAT(3)) GO TO 2191
            CPTR=SRUPTR(I2181)
            TC1=BRLDS(CPTR)
            TC2=BRLSS(CPTR)
            W=SDARC(CPTR)
            CAP(W)=CAP(W)+TC1+TC2
            X=SSARC(CPTR)
            CAP(X)=CAP(X)+TC2
           IPTR=SBSARC(CPTR)
           SRCE(IPTR)=LNODE+10
           IPTR=SBDARC(CPTR)
           SRCE(IPTR)=LNODE+10
     2191 DO 2182 I2182=1,NARCS
           IF(DEST(I2182).EQ.(LDSE+I2181)) SRCE(I2182)=LNODE+10
          IF(SRCE(I2182).EQ.(LDSE+I2181)) SRCE(I2182)=LNODE+10
           IF(SRCE(I2182).EQ.(LDSRU+I2181)) SRCE(I2182)=LNODE+10
           IF(DEST(I2182).EQ.(LDSRU+I2181)) SRCE(I2182)=LNODE+10
     2182   CONTINUE
     2181  CONTINUE
          DO 2184 I2184=1,SFMS
           IF(OPDECS(I2184).EQ.LOCAT(15)) GO TO 2184
             IPTR=SDARC(I2184)
             SRCE(IPTR)=LNODE+10
             IPTR=SBARC(I2184)
             SRCE(IPTR)=LNODE+10
```

```
          IPTR=SSARC(I2184)
          SRCE(IPTR)=LNODE+10
          DO 2185 I2185=1,NARCS
        IF(DEST(I2185).EQ.(LBLRU+I2184)) SRCE(I2185)=LNODE+10
        IF(SRCE(I2185).EQ.(LBLRU+I2184)) SRCE(I2185)=LNODE+10
        IF(DEST(I2185).EQ.(LBLRU+I2184)) SRCE(I2185)=LNODE+10
 2185    CONTINUE
 2184   CONTINUE
C       EXPLICITLY SORT THE SRCE-DEST-CAP TRIPLES INTO ASCENDING ORDER
C       BY SRCE (FLOW IS USED BY SORT TO STORE ORDERING POINTERS)
       CALL SORT(SRCE,DEST,CAP,FLOW,NARCS,MAXARC,1)
       ITER=0
       DO 2487 I2487=1,NARCS
       IF(SRCE(I2487).LT.LNODE) ITER=ITER+1
 2487  CONTINUE
       NARCS=ITER
C       SAVE POINTERS FOR FORWARD SCAN IN FWDSP
       DO 2490 I2490=1,NARCS
C       UTILIZE THE VALUES IN SRCE IN REVERSE ORDER.  (IF NODE '1' IS A
C       SOURCE FOR 'N' ARCS THEN FWDSP(1) WILL SUCCESSIVELY GET THE
C       VALUES: N, N-1, N-2, . . . , 2, 1.  SIMILARLY, FWDSP(K) WILL
C       HAVE THE VALUE 'J' WHERE J CORRESPONDS TO THE FIRST OCCURENCE
C       OF NODE # K IN SRCE.)
        J2490=NARCS+1-I2490
        K2490=SRCE(J2490)
        FWDSP(K2490)=J2490
 2490   CONTINUE
       FWDSP(LNODE)=NARCS+1
       LNODM1=LNODE-1
       DO 2495 I2495=1,LNODM1
       J2495=LNODE-I2495
       IF(FWDSP(J2495).EQ.0)  FWDSP(J2495)=FWDSP(J2495+1)
 2495  CONTINUE
C
C       USE SORT TO IMPLICITLY SEQUENCE THE DEST ENTRIES INTO ASCENDING
C       ORDER.  THE ORDER WILL BE SPECIFIED BY THE ENTRIES IN BKPTR.
C       THUS, THE ENTRIES IN BACKSP ARE POINTERS TO SRCE-DEST-CAP
C       TRIPLES BASED ON THE DEST VALUES.
       CALL SORT(DEST,0,0,BKPTR,NARCS,MAXARC,0)
       BACKSP(1)=0
       DO 2500 I2500=1,NARCS
       J2500=BKPTR(I2500)
       K2500=DEST(J2500)
C       BACKSP(K) IS THE POSITION IN BKPTR WHICH HOLDS THE LAST POINTER
C       TO A SRCE-DEST-CAP TRIPLE HAVING NODE # K AS THE DEST.  THE FIRST
C       POINTER TO A TRIPLE HAVING NODE # K AS THE DEST IS BKPTR(K-1)+1.
       BACKSP(K2500)=I2500
 2500  CONTINUE
       DO 2525 I2525=2,LNODE
       IF(BACKSP(I2525).EQ.0)  BACKSP(I2525)=BACKSP(I2525-1)
 2525  CONTINUE
C
       DO 2600 I2600=1,NARCS
```

```
      J=SPCE(I2600)
      K=DEST(I2600)
      IF(J.NE.1)  GO TO 2530
      IF(K.GT.LDSE)  GO TO 2510
      SEARCP(K-1)=I2600
      GO TO 2595
 2510 IF(K.GT.LDLRU)  GO TO 2520
      LDARC(K-LDSE)=I2600
      GO TO 2595
 2520 SDARC(K-LDLRU)=I2600
      GO TO 2595
 2530 IF(J.LE.LDSE)  GO TO 2595
      IF(J.GT.LDLRU)  GO TO 2550
      IF(K.LE.LDSRU)  GO TO 2540
      LSARC(J-LDSE)=I2600
      GO TO 2595
 2540 SBDARC(K-LDLRU)=I2600
      GO TO 2595
 2550 IF(J.GT.LDSRU)  GO TO 2560
      SSARC(J-LDLRU)=I2600
      GO TO 2595
 2560 IF(J.GT.LBLRU)  GO TO 2580
      IF(K.NE.LNODE)  GO TO 2570
      LBARC(J-LDSRU)=I2600
      GO TO 2595
 2570 IF(K.GT.LBSRU)  GO TO 2595
      SBSARC(K-LBLRU)=I2600
      GO TO 2595
 2580 IF(J.GT.LBSRU)  GO TO 2590
      IF(K.NE.LNODE)  GO TO 2595
      SBARC(J-LBLRU)=I2600
      GO TO 2595
 2590 SEARCP(J+NDSE-LBSRU)=I2600
 2595 FLOW(I2600)=0
 2600 CONTINUE
 3000 CONTINUE
C IF(MSGL2.LT.2)  GO TO 2440
C DO 2410 I=1,LFMS
C PRINT,'LPTRS ',LDARC(I),LSARC(I),LBARC(I)
C 2410 CONTINUE
C DO 2420 I=1,SFMS
C PRINT,'SPTRS ',SDARC(I),SSARC(I),SBARC(I),SBDARC(I),SBSARC(I)
C 2420 CONTINUE
C DO 2430 I=1,NUMSER
C PRINT,'SEPTRS ',SEARCP(I)
C 2430 CONTINUE
C 2440 CONTINUE
      RETURN
      END
```

Appendix F: NMXFLO Subroutine

```
      SUBROUTINE NMXFLO(MSGL2)
C
C
C      TEMPORARY DATA INPUT AREAS
       INTEGER MIRROR,NSTART,IAPREV,NAPATH(300)
C
      IF (ICIRF.EQ.2) GOTO 3012
      DO 3005 I3005=1,LFMS
      IF (OPDECL(I3005).NE.LOCAT(15)) GOTO 3005
       IF(LFMOD(I3005).EQ.0)  GO TO 3001
       J3005=LDARC(I3005)
       CAP(J3005)=JUMBO
 3001  IF(LFMOS(I3005).EQ.0)  GO TO 3002
       J3005=LSARC(I3005)
       CAP(J3005)=JUMBO
 3002  IF(LFMOB(I3005).EQ.0)  GO TO 3005
       J3005=LBARC(I3005)
       CAP(J3005)=JUMBO
 3005  CONTINUE
C
       IF(SFMS.LT.1)  GO TO 3012
       DO 3010 I3010=1,SFMS
       IF (OPDECS(I3010).NE.LOCAT(15)) GOTO 3010
       IF(SOD(I3010).EQ.0)  GO TO 3006
       J3010=SDARC(I3010)
       CAP(J3010)=JUMBO
 3006  IF(SOS(I3010).EQ.0)  GO TO 3007
       J3010=SSARC(I3010)
       CAP(J3010)=JUMBO
 3007  IF(SOB(I3010).EQ.0)  GO TO 3010
       J3010=SBARC(I3010)
       CAP(J3010)=JUMBO
 3010  CONTINUE
 3012  CONTINUE
       CALL PTIME(X)
C
C     DEPTH FIRST MAX FLOW ALGOTITHM
C
C     DATA USED IS CONTAINED IN THE VECTORS SRCE, DEST ,CAP, FLOW
C     AND BKPTR WHICH HAVE AN ENTRY FOR EACH ARC; AND IN VECTORS
C     NPATH, DLTAFL, STATE, FWDSP, AND BACKSP WHICH HAVE AN ENTRY FOR
C     EACH NODE.  STATE IS A '1' FOR AN UNLABELED NODE AND A '2' FOR A
C     LABELED NODE.
C
C     LABEL THE SOURCE
C
        DLTAFL(1) = JUMBO
        MIRROR = 0
        NPATH(1) = -1
 5015   STATE(1) = 2
```

```
C
C     LABEL ALL OTHER NODES AS UNLABELED '1'
C
       DO 5020 I5020 = 2,LNODE
        STATE(I5020) = 1
 5020   CONTINUE
C
C      FIND AN ARC FROM NODE 1 WITH CAPACITY REMAINING
C
        J5050 = FWDSP(1)
        K5050 = FWDSP(2) - 1
        DO 5060 I5060 = J5050,K5050
        NODE = DEST(I5060)
        IF (FLOW(I5060).LT.CAP(I5060).AND.STATE(NODE).EQ.1) GOTO 5065
        GOTO 5060
C
C     MARK FORWARD OPATH AND CALC MAX FLOW ON ARC I5060
C
 5065   NPATH(NODE) = 1
        STATE(NODE) = 2
        NAPATH(NODE) = I5060
        JDELTA = CAP(I5060) - FLOW(I5060)
        DLTAFL(NODE) = MIN0(JDELTA,DLTAFL(1))
C
C     CONTINUE FORWARD DEPTH FIRST FLOW AUGMENTATION
C
 5070   NNODE = NODE
        J5080 = FWDSP(NNODE)
        K5080 = FWDSP(NNODE + 1) - 1
        DO 5090 I5090 = J5080,K5080
        NODE = DEST(I5090)
        IF (FLOW(I5090).LT.CAP(I5090).AND.STATE(NODE).EQ.1) GOTO 5085
        GOTO 5090
 5085   NPATH(NODE) = NNODE
        STATE(NODE) = 2
        NAPATH(NODE) = I5090
        JDELTA = CAP(I5090) - FLOW(I5090)
        DLTAFL(NODE) = MIN0(JDELTA,DLTAFL(NNODE))
C
C     IF NODE EQUALS LAST NODE INCREMENT FLOW ON PATH
C
        IF (NODE.EQ.LNODE) GOTO 5150
        GOTO 5070
 5090   CONTINUE
C
C      FIRST TIME THROUGH SKIP MIRROR ARC SECTION
C
        IF (MIRROR.EQ.0) GOTO 5140
C
C      MIRROR ARC AUGMENTING PATH SECTION
C
 5100    J5100 = BACKSP(NNODE - 1) + 1
         J5110 = BACKSP(NNODE)
```

```
          DO 5130 I5130 = J5100,J5110
          K5100 = BKPTR(I5130)
          NODE = SRCE(K5100)
C
C     CHECK FOR SOURCE NODE OR PATH JUST TRAVELED
C
          IF (NODE.EQ.1.OR.NODE.EQ.NPATH(NNODE)) GOTO 5130
          IF (FLOW(K5100).GT.0.AND.STATE(NODE).EQ.1) GOTO 5120
          GOTO 5130
C
C     MARK REVERSE PATH AND CALC MAX FLOW ON ARC K5100
C
     5120 NPATH(NODE) = 0 - NNODE
          STATE(NODE) = 2
          NAPATH(NODE) = K5100
          DLTAFL(NODE) = MIN0(FLOW(K5100),DLTAFL(NNODE))
          GOTO 5070
     5130 CONTINUE
C
C     IF UNABLE TO AUGMENT PATH TO SINK FROM THIS NODE
C     LABEL IT '2' AND BACKUP TO PREVIOUS NODE ON PATH
C
     5140 NODE = IABS(NPATH(NNODE))
          IF (NODE.EQ.1) GOTO 5060
          GOTO 5070
C
C     FLOW AUGMENTATION ON PATH FROM SOURCE TO SINK
C
     5150 LAST = LNODE
          STATE(LAST) = 1
          NSTART = 0
          INC = DLTAFL(LNODE)
     5160 IPREV = NPATH(LAST)
          IAPREV = NAPATH(LAST)
          IF (IPREV.GT.0) GOTO 5170
          FLOW(IAPREV) = FLOW(IAPREV) - INC
          GOTO 5180
     5170 FLOW(IAPREV) = FLOW(IAPREV) + INC
     5180 LAST = IABS(IPREV)
          IF (LAST.EQ.1) GOTO 5200
          DLTAFL(LAST) = DLTAFL(LAST) - INC
C
C     IDENTIFY AND MARK THE LAST NODE IN THE PATH WITH EXCESS
C     FLOW REMAINING FOR AUGMENTATION TO THE SINK
C
         IF (FLOW(IAPREV).LT.CAP(IAPREV).AND.DLTAFL(LAST).EQ.0)
        & STATE(LAST) = 1
          IF (NSTART.EQ.1.OR.DLTAFL(LAST).EQ.0) GOTO 5190
          NSTART = 1
          NODE = LAST
     5190 GOTO 5160
```

```
 5200  IF (NSTART.EQ.1) GOTO 5070
 5060  CONTINUE
C
C    AFTER SECOND TIME THROUGH SOLUTION COMPLETE
C
       IF (MIRROR.EQ.2) GOTO 3300
C
C    SET SWITCH TO ALLOW FOR FLOW AUGMENTATION ON MIRROR ARCS
C    AND START OVER
C
       MIRROR = MIRROR + 1
        GOTO 5015
C
C
C       ****  NETWORK SOLUTION COMPLETE  ****
C       COMPUTE MAX FLOW
 3300 J3310=1
      K3310=FWDSP(2)-1
      ITFLOW=0
      DO 3310 I3310=J3310,K3310
      ITFLOW=ITFLOW+FLOW(I3310)
 3310 CONTINUE
      FLOW(MAXARC+1)=ITFLOW
C IF(MSGL2.LT.1)  GO TO 3400
C PRINT,' TOTAL FLOW ',ITFLOW
C PRINT 2220,'NODE #',(I,I=1,LNODE)
C PRINT 2220,'SRCE  ',(SRCE(I),I=1,NARCS)
C PRINT 2220,'DEST  ',(DEST(I),I=1,NARCS)
C PRINT 2220,'NPATH ',(NPATH(I),I=1,LNODE)
C PRINT 2220,'DLTAFL',(DLTAFL(I),I=1,LNODE)
C PRINT 2220,'STATE ',(STATE(I),I=1,LNODE)
C PRINT 2220,'FLOW  ',(FLOW(I),I=1,NARCS)
C PRINT 2220,'CAP   ',(CAP(I),I=1,NARCS)
C 3400 CONTINUE
      LCSEC=0
      NDSEP1=NDSE+1
      IF(NDSE.EQ.0)  GO TO 3330
      DO 3320 I3320=2,NDSEP1
      IF(STATE(I3320).NE.1)  GO TO 3320
      SEPTR=SEARCP(I3320-1)
      LCSEC=LCSEC+CAP(SEPTR)
 3320 CONTINUE
 3330 IF(NDSE.EQ.NUMSER)  GO TO 3350
      DO 3340 I3340=NDSEP1,NUMSER
      J3340=LBSRU+I3340-NDSE
      IF(STATE(J3340).EQ.1)  GO TO 3340
      SEPTR=SEARCP(I3340)
      LCSEC=LCSEC+CAP(SEPTR)
 3340 CONTINUE
 3350 FLOW(MAXARC+2)=LCSEC
C J=0
```

```
C DO 3360 I=1,NARCS
C IF(DEST(I).EQ.LNODE)  J=J+FLOW(I)
C 3360 CONTINUE
C IF(J.NE.ITFLOW)  PRINT,'NOT CONSISTENT FLOW ',ITFLOW,J
        Y = 0
        CALL PTIME(Y)
        Z=Y-X
        TT=TT+Z
       WRITE(6,400) X*3600,Y*3600,Z*3600,TT*3600
  400  FORMAT(' TIME=',4F10.5)
       RETURN
       END

*
```

Appendix G: OUTPUT Subroutine

```
      SUBROUTINE OUTPUT
C
C
C
C      OUTPUT RESULTS FOR DEPOT SE
 3400 IPAGE=IPAGE+1
      WRITE(6,3401)  (SYSNAM(ISUB),ISUB=1,3),DATE,IPAGE
 3401 FORMAT('1',2X,3A4,86X,A8,5X,'PAGE',I4)
      WRITE(6,3402)
 3402 FORMAT(' ',24X,'NETWORK RLA RESULTS')
      WRITE(6,3403)
 3403 FORMAT(' ',90X,'WHOLESALE CHANGE FACTORS')
      WRITE(6,3404) WCFUC,WCFF,WCFSE
 3404 FORMAT(' ',96X,'COST =',F6.2/97X,'MTBF =',F6.2/99X,'SE =',F6.2)
      WRITE(6,3410)
 3410 FORMAT(' ',22X,'SUPPORT EQUIPMENT REQUIREMENTS')
      WRITE(6,3412)
 3412 FORMAT('0',4X,'DEPOT',28X,'TOTAL SE')
      WRITE(6,3415)
 3415 FORMAT(' ','SE CODE',5X,'NAME',10X,'REQMT',5X,'LC $/BASE',3X,
     &    'USE HRS',7X,'USE %')
      TOTSED=0.
      IF(NDSE.LT.1)  GO TO 3431
      DO 3430 I3430=1,NDSE
      IF(STATE(I3430+1).GT.1)  GO TO 3430
      WRITE(6,3420)  SECODE(I3430),(XSE(I3430,ISUB),ISUB=1,3),
     & REQMT(I3430),TSECST(I3430),USEHRS(I3430),SEUP(I3430)
 3420 FORMAT(' ',2X,I5,5X,3A4,3X,I3,1X,F12.0,F12.0,2PF12.1)
      TOTSED=TOTSED+TSECST(I3430)
 3430 CONTINUE
 3431 CONTINUE
      WRITE(6,3435) TOTSED
 3435 FORMAT('0',2X,'TOTAL',24X,F12.0)
C
C      OUTPUT RESULTS FOR BASE SUPPORT EQUIPMENT
      IF(NDSE.EQ.NUMSER)  GO TO 3451
      WRITE(6,3440)
 3440 FORMAT('0',4X,'BASE',29X,'TOTAL SE')
      WRITE(6,3415)
      TOTSEB=0.
      NDSEP1=NDSE+1
      NBSEP0=NDSE+NBSE
      DO 3450 I3450=NDSEP1,NBSEP0
      J3450=LBSPU+I3450-NDSE
      IF(STATE(J3450).EQ.1)  GO TO 3450
 3445 WRITE(6,3420)  SECODE(I3450),(XSE(I3450,ISUB),ISUB=1,3),
     &    REQMT(I3450),SECOST(I3450),USEHRS(I3450),SEUR(I3450)
      TOTSEB=TOTSEB+TSECST(I3450)
 3450 CONTINUE
      WRITE(6,3435) TOTSEB
```

```
C      OUTPUT RESULTS FOR CIRF SUPPORT EQUIPMENT
 3442 IF (NBSEPO.EQ.NUMSER) GOTO 3451
      WRITE(6,3800)
 3800 FORMAT('0',4X,'CIRF',29X,'TOTAL SE')
      WRITE (6,3415)
      IF (ICIRF.EQ.1) WRITE (6,3810)
 3810 FORMAT('    NO CIRF SE CALCULATED THIS PASS')
      IF (ICIRF.EQ.1) GOTO 3451
      TOTSEC=0
      NBSEP1=NBSEPO+1
      DO 3820 I3820=NBSEP1,NUMSER
      J3820=I3820-NBSEPO
      IF (STATE(J3820).GT.1) GOTO 3820
      WRITE (6,3420) SECODE(I3820),(XSE(I3820,ISUB),ISUB=1,3),
     & REQMT(I3820),TSECST(I3820),USEHRS(I3820),SEUR(I3820)
      TOTSEC=TOTSEC+TSECST(I3820)
 3820 CONTINUE
      WRITE (6,3435) TOTSEC
C
C      OUTPUT RESULTS FOR LRU'S & SRU'S
 3451 IPAGE=IPAGE+1
      WRITE(6,3401)  (SYSNAM(ISUB),ISUB=1,3),DATE,IPAGE
      WRITE(6,3452)  CHARS(2),CHARS(2)
 3452 FORMAT(' ',40X,'LRU & SRU REPAIR LEVEL DECISIONS',2X,2A4)
      WRITE(6,3453)
 3453 FORMAT('0',29X,'-- LRU (LC $/BASE) --',7X,'-- SRU ',
     &    '(LC $/BASE) --',7X,'LRU/SRU   LRU',7X,'INPUT     MTB      ',
     &    'LC REP'/' ',4X,'LRU/SRU NAME  IDENT NO   DEPOT   SCRAP',
     &    '     BASE',' CIRF','DEPOT   SCRAP    BASE',' CIRF     ',
     &    'COST  FAIL%',7X,'MTBF     REPAIR  DEM/BASE')
      TOTLD=0.
      TOTLS=0.
      TOTLC=0.
      TOTLB=0.
      TOTSD=0.
      TOTSS=0.
      TOTSB=0.
      LINES=3
      WUC(1)=LOCAT(15)
      WUC(2)=LOCAT(15)
      NDSEP2=NDSE+2
      NSE=NDSEP2
      ICIRF=1
      LLRU=LDLRU
      DO 3600 I3600=NSE,LLRU
      ITEM=I3600-(NSE-1)
      IF(LEMWUC(ITEM,1).EQ.WUC(1).AND.LEMWUC(ITEM,2).EQ.WUC(2))
     &    GO TO 3457
      DO 3454 I3454=1,NLRU
       IF(LEMWUC(ITEM,1).EQ.LWUC(I3454,1).AND.
     &    LEMWUC(ITEM,2).EQ.LWUC(I3454,2))  LPTR=I3454
 3454 CONTINUE
       MTBCT=MTBF(LPTR)/(UF(LPTR)*(1.-RIP(LPTR)))
```

```
      TOCTCM=PCMB*OPA(LPTR)/MTBCT
      TLCD=TOCTCM*PIUP*12.
 3457 IF(LINES.LT.54)  GO TO 3458
      IPAGE=IPAGE+1
      WRITE(6,3401)  (SYSNAM(ISUB),ISUB=1,3),DATE,IPAGE
      WRITE(6,3452)  CHARS(5),CHARS(6)
      WRITE(6,3453)
      LINES=4
 3458 TLCDF=TLCD*FAILP(ITEM)
      FMTBCT=MTBCT/FAILP(ITEM)
      NODEL=I3600
      NODER=NODEL+LFMS+SFMS
      IF (OPDECL(ITEM).EQ.LOCAT(4)) GOTO 3483
      IF (OPDECL(ITEM).EQ.LOCAT(1)) GOTO 3460
      IF (OPDECL(ITEM).EQ.LOCAT(2)) GOTO 3470
      IF (OPDECL(ITEM).EQ.LOCAT(3)) GOTO 3480
C      CHECK FOR LRU DEPOT REPAIR
      IF(STATE(NODEL).EQ.1.AND.STATE(NODER).EQ.1)  GO TO 3460
C      CHECK FOR LRU SCRAPPED
      IF(STATE(NODEL).NE.1.AND.STATE(NODER).EQ.1)  GO TO 3470
C      CHECK FOR LRU BASE REPAIRED
      IF(STATE(NODEL).NE.1.AND.STATE(NODER).NE.1)  GO TO 3480
      WRITE(6,3459)
 3459 FORMAT(' TROUBLE ON LRU LABELS')
      STOP
C
 3460 COSTL=DEPOLC(ITEM)
      TOTLD=TOTLD+COSTL
      LEVEL=1
      OPDECL(ITEM)=LOCAT(1)
      GO TO 3490
C
 3470 COSTL=SCRPLC(ITEM)
      TOTLS=TOTLS+COSTL
      LEVEL=2
      OPDECL(ITEM)=LOCAT(2)
      GO TO 3490
C
 3480 COSTL=BASELC(ITEM)
      TOTLB=TOTLB+COSTL
      LEVEL=3
      OPDECL(ITEM)=LOCAT(3)
      GOTO 3490
 3483 COSTL=CIRFLC(ITEM)
      TOTLC=TOTLC+COSTL
      LEVEL=4
      OPDECL(ITEM)=LOCAT(4)
 3490 FPCT=FAILP(ITEM)*100.
      LOD=CHARS(2)
      LOS=CHARS(2)
      LOB=CHARS(2)
      LOC=CHARS(2)
      IF(LEMOD(ITEM).EQ.2)  LOD=CHARS(14)
```

```
      IF(LFMOS(ITEM).EQ.2)  LOS=CHARS(14)
      IF(LFMOB(ITEM).EQ.2)  LOB=CHARS(14)
      IF (LFMOC(ITEM).EQ.2) LOC=CHARS(14)
      IF(SRUPTR(ITEM).NE.0)  GO TO 3520
      GO TO (3493,3500,3510,3580),LEVEL
 3493 IF(WUC(1).NE.LFMWUC(ITEM,1).OR.WUC(2).NE.LFMWUC(ITEM,2))
     &     GO TO 3496
 3494 WRITE(6,3495)  COSTL,LOS,LOB,LOC,FPCT,FMTBCT,TLCDF
 3495 FORMAT(' ',27X,F8.0,3X,A4,4X,A4,2X,A4,38X,F5.1,13X,F8.0,2X,F8.1)
      LINES=LINES+1
      GO TO 3600
 3496 IF(FRSTFM(LPTR).EQ.LASTFM(LPTR))  GO TO 3498
      WRITE(6,3497)  LPTR,LRUNAM(LPTR,1),LRUNAM(LPTR,2),LRUNAM(LPTR,3),
     &     LWUC(LPTR,1),LWUC(LPTR,2),UCL(LPTR),MTBF(LPTR),MTBCT,TLCD
 3497 FORMAT('0',I2,2X,3A4,2X,A4,A3,59X,F8.0,10X,F8.0,2X,F8.0,
     &     2X,F8.1)
      LINES=LINES+2
      WUC(1)=LFMWUC(ITEM,1)
      WUC(2)=LFMWUC(ITEM,2)
      GO TO 3494
 3498 WRITE(6,3499)  LPTR,LRUNAM(LPTR,1),LRUNAM(LPTR,2),LRUNAM(LPTR,3),
     &     LWUC(LPTR,1),LWUC(LPTR,2),COSTL,LOS,LOB,LOC,UCL(LPTR),
     &     FPCT,MTBF(LPTR),FMTBCT,TLCDF
 3499 FORMAT('0',I2,2X,3A4,2X,A4,A3,2X,F8.0,3X,A4,4X,A4,2X,A4,28X,F8.0,
     &     2X,F5.1,3X,F8.0,2X,F8.0,2X,F8.1)
      LINES=LINES+2
      WUC(1)=LFMWUC(ITEM,1)
      WUC(2)=LFMWUC(ITEM,2)
      GO TO 3600
C
 3500 IF(WUC(1).NE.LFMWUC(ITEM,1).OR.WUC(2).NE.LFMWUC(ITEM,2))
     &     GO TO 3503
 3501 WRITE(6,3502)  LOD,COSTL,LOB,LOC,FPCT,FMTBCT,TLCDF
 3502 FORMAT(' ',30X,A4,1X,F8.0,3X,A4,2X,A4,38X,F5.1,13X,F8.0,2X,F8.1)
      LINES=LINES+1
      GO TO 3600
 3503 IF(FRSTFM(LPTR).EQ.LASTFM(LPTR))  GO TO 3504
      WRITE(6,3497)  LPTR,LRUNAM(LPTR,1),LRUNAM(LPTR,2),LRUNAM(LPTR,3),
     &     LWUC(LPTR,1),LWUC(LPTR,2),UCL(LPTR),MTBF(LPTR),MTBCT,TLCD
      LINES=LINES+2
      WUC(1)=LFMWUC(ITEM,1)
      WUC(2)=LFMWUC(ITEM,2)
      GO TO 3501
 3504 WRITE(6,3505)  LPTR,LRUNAM(LPTR,1),LRUNAM(LPTR,2),LRUNAM(LPTR,3),
     &     LWUC(LPTR,1),LWUC(LPTR,2),LOD,COSTL,LOB,LOC,
     &     UCL(LPTR),FPCT,MTBF(LPTR),FMTBCT,TLCDF
 3505 FORMAT('0',I2,2X,3A4,2X,A4,A3,5X,A4,1X,F8.0,3X,A4,2X,A4,28X,
     &     F8.0,2X,F5.1,3X,F8.0,2X,F8.0,2X,F8.1)
      LINES=LINES+2
      WUC(1)=LFMWUC(ITEM,1)
      WUC(2)=LFMWUC(ITEM,2)
      GO TO 3600
C
```

```
3510 IF(WUC(1).NE.LFMWUC(ITEM,1).OR.WUC(2).NE.LFMWUC(ITEM,2))
     &    GO TO 3513
3511 WRITE(6,3512)  LOD,LOS,COSTL,LOC,FPCT,FMTBCT,TLCDF
3512 FORMAT(' ',30X,A4,4X,A4,1X,F8.0,1X,A4,38X,F5.1,13X,F8.0,2X,F8.1)
      LINES=LINES+1
      GO TO 3600
3513 IF(FRSTFM(LPTR).EQ.LASTFM(LPTR))  GO TO 3514
      WRITE(6,3497)  LPTR,LRUNAM(LPTR,1),LRUNAM(LPTR,2),LRUNAM(LPTR,3),
     &    LWUC(LPTR,1),LWUC(LPTR,2),UCL(LPTR),
     &    MTBF(LPTR),MTBCT,TLCD
      LINES=LINES+2
      WUC(1)=LFMWUC(ITEM,1)
      WUC(2)=LFMWUC(ITEM,2)
      GO TO 3511
3514 WRITE(6,3515)  LPTR,LRUNAM(LPTR,1),LRUNAM(LPTR,2),LRUNAM(LPTR,3),
     &    LWUC(LPTR,1),LWUC(LPTR,2),LOD,LOS,COSTL,LOC,
     &    UCL(LPTR),FPCT,MTBF(LPTR),FMTBCT,TLCDF
3515 FORMAT('0',I2,2X,3A4,2X,A4,A3,5X,A4,4X,A4,1X,F8.0,1X,A4,28X,F8.0,2X,
     &    F5.1,3X,F8.0,2X,F8.0,2X,F8.1)
      LINES=LINES+2
      WUC(1)=LFMWUC(ITEM,1)
      WUC(2)=LFMWUC(ITEM,2)
      GO TO 3600
C
3580 IF(WUC(1).NE.LFMWUC(ITEM,1).OR.WUC(2).NE.LFMWUC(ITEM,2))
     &    GO TO 3583
3581 WRITE(6,3582)  LOD,LOS,LOB,COSTL,FPCT,FMTBCT,TLCDF
3582 FORMAT(' ',30X,A4,3X,A4,3X,A4,1X,F7.0,38X,F5.1,13X,F8.0,2X,F8.1)
      LINES=LINES+1
      GO TO 3600
3583 IF(FRSTFM(LPTR).EQ.LASTFM(LPTR))  GO TO 3584
      WRITE(6,3497)  LPTR,LRUNAM(LPTR,1),LRUNAM(LPTR,2),LRUNAM(LPTR,3),
     &    LWUC(LPTR,1),LWUC(LPTR,2),UCL(LPTR),
     &    MTBF(LPTR),MTBCT,TLCD
      LINES=LINES+2
      WUC(1)=LFMWUC(ITEM,1)
      WUC(2)=LFMWUC(ITEM,2)
      GO TO 3581
3584 WRITE(6,3585)  LPTR,LRUNAM(LPTR,1),LRUNAM(LPTR,2),LRUNAM(LPTR,3),
     &    LWUC(LPTR,1),LWUC(LPTR,2),LOD,LOS,LOB,COSTL,
     &    UCL(LPTR),FPCT,MTBF(LPTR),FMTBCT,TLCDF
3585 FORMAT('0',I2,2X,3A4,2X,A4,A3,5X,A4,3X,A4,3X,A4,1X,F7.0,29X,
     &    F8.0,2X,F5.1,3X,F8.0,2X,F8.0,2X,F8.1)
      LINES=LINES+2
      WUC(1)=LFMWUC(ITEM,1)
      WUC(2)=LFMWUC(ITEM,2)
      GO TO 3600
C
3520 IF(WUC(1).EQ.LFMWUC(ITEM,1).AND.WUC(2).EQ.LFMWUC(ITEM,2))
     &    GO TO 3525
      WRITE(6,3497)  LPTR,LRUNAM(LPTR,1),LRUNAM(LPTR,2),LRUNAM(LPTR,3),
     &    LWUC(LPTR,1),LWUC(LPTR,2),UCL(LPTR),
     &    MTBF(LPTR),MTBCT,TLCD
```

```
      LINES=LINES+2
      WUC(1)=LEMWUC(ITEM,1)
      WUC(2)=LEMWUC(ITEM,2)
C
 3525 NODEL=LDLRU+SRUPTR(ITEM)
      NODER=LBLRU+SRUPTR(ITEM)
      ITEM=SRUPTR(ITEM)
      ISOD=CHARS(2)
      ISOS=CHARS(2)
      ISOB=CHARS(2)
      ISOC=CHARS(2)
      IF(SOD(ITEM).EQ.2)  ISOD=CHARS(14)
      IF(SOS(ITEM).EQ.2)  ISOS=CHARS(14)
      IF(SOB(ITEM).EQ.2)  ISOB=CHARS(14)
      IF(SOC(ITEM).EQ.2) ISOC=CHARS(14)
      IF (OPDECS(ITEM).EQ.LOCAT(4)) GOTO 3680
      IF (ICIRF.EQ.2.AND.STATE(NODEL).EQ.1) GOTO 3680
      IF (OPDECS(ITEM).EQ.LOCAT(1)) GOTO 3530
      IF (OPDECS(ITEM).EQ.LOCAT(2)) GOTO 3550
      IF (OPDECS(ITEM).EQ.LOCAT(3)) GOTO 3570
C      CHECK FOR SRU DEPOT REPAIR
      IF(STATE(NODEL).EQ.1.AND.STATE(NODER).EQ.1)  GO TO 3530
C      CHECK FOR SRU SCRAPPED
      IF(STATE(NODEL).NE.1.AND.STATE(NODER).EQ.1)  GO TO 3550
C      CHECK FOR SRU BASE REPAIRED
      IF(STATE(NODEL).NE.1.AND.STATE(NODER).NE.1)  GO TO 3570
      WRITE(6,3526)
 3526 FORMAT(' TROUBLE ON SRU LABELS')
      STOP
C      SRU IS DEPOT REPAIRED
 3530 COST=DEPOSC(ITEM)
C      IF LRU IS BASE REPAIRED ADD EXTRA COSTS
      IF(LEVEL.EQ.3)  COST=COST+BRLDS(ITEM)+BRLSS(ITEM)
C
      TOTSD=TOTSD+COST
      OPDECS(ITEM)=LOCAT(1)
      GO TO (3532,3534,3536,3538),LEVEL
 3532 WRITE(6,3533)  SRUNAM(ITEM,1),SRUNAM(ITEM,2),SWUC(ITEM,1),
     &    SWUC(ITEM,2),COSTL,LOS,LOB,LOC,COST,ISOS,ISOB,ISOC,UCS(ITEM),
     &    FPCT,FMTBCT,TLCDF
 3533 FORMAT(' ',8X,2A4,2X,A4,A3,2X,F8.0,3X,A4,4X,A4,1X,A4,F8.0,
     &    3X,A4,4X,A4,2X,A4,F8.0,2X,F5.1,13X,F8.0,2X,F8.1)
      GO TO 3590
 3534 WRITE(6,3535)  SRUNAM(ITEM,1),SRUNAM(ITEM,2),SWUC(ITEM,1),
     &    SWUC(ITEM,2),LOD,COSTL,LOB,LOC,COST,ISOS,ISOB,ISOC,UCS(ITEM),
     &    FPCT,FMTBCT,TLCDF
 3535 FORMAT(' ',3X,2A4,2X,A4,A3,5X,A4,1X,F8.0,3X,A4,1X,A4,F8.0,
     &    3X,A4,4X,A4,2X,A4,F8.0,2X,F5.1,
     &    13X,F8.0,2X,F8.1/40X,'CASE 7 ERROR IN ABOVE DECISION')
      GO TO 3590
 3536 WRITE(6,3537)  SRUNAM(ITEM,1),SRUNAM(ITEM,2),SWUC(ITEM,1),
     &    SWUC(ITEM,2),LOD,LOS,COSTL,LOC,COST,ISOS,ISOB,ISOC,UCS(ITEM),
     &    FPCT,FMTBCT,TLCDF
```

```
 3537 FORMAT(' ',8X,2A4,2X,A4,A3,5X,A4,4X,A4,1X,F7.0,1X,A4,F8.0,
     &    3X,A4,4X,A4,2X,A4,F8.0,2X,F5.1,
     &    13X,F8.0,2X,F8.1)
      GO TO 3590
 3538 WRITE(6,3539)  SRUNAM(ITEM,1),SRUNAM(ITEM,2),SWUC(ITEM,1),
     &    SWUC(ITEM,2),LOD,LOS,LOB,COSTL,COST,ISOS,ISOB,ISOC,UCS(ITEM),
     &    FPCT,FMTBCT,TLCDF
 3539 FORMAT(' ',8X,2A4,2X,A4,A3,5X,A4,4X,A4,2X,A4,F7.0,F8.0,
     &    3X,A4,4X,A4,2X,A4,F8.0,2X,F5.1,13X,F8.0,2X,F8.1)
      GO TO 3590
C
C      SRU IS SCRAPPED
 3550 COST=SCRPSC(ITEM)
C      IF LRU IS BASE REPAIRED ADD EXTRA COST
      IF(LEVEL.EQ.3)  COST=COST+BRLSS(ITEM)
      TOTSS=TOTSS+COST
      OPDECS(ITEM)=LOCAT(2)
      GO TO (3552,3554,3556,3558),LEVEL
 3552 WRITE(6,3553)  SRUNAM(ITEM,1),SRUNAM(ITEM,2),SWUC(ITEM,1),
     &    SWUC(ITEM,2),COSTL,LOS,LOB,LOC,ISOD,COST,ISOB,ISOC,UCS(ITEM),
     &    FPCT,FMTBCT,TLCDF
 3553 FORMAT(' ',8X,2A4,2X,A4,A3,2X,F8.0,3X,A4,4X,A4,2X,A4,2X,A4,1X,
     &    F8.0,3X,A4,2X,A4,F8.0,2X,F5.1,
     &    13X,F8.0,2X,F8.1)
      GO TO 3590
 3554 WRITE(6,3555)  SRUNAM(ITEM,1),SRUNAM(ITEM,2),SWUC(ITEM,1),
     &    SWUC(ITEM,2),LOD,COSTL,LOB,LOC,ISOD,COST,ISOB,ISOC,UCS(ITEM),
     &    FPCT,FMTBCT,TLCDF
 3555 FORMAT(' ',8X,2A4,2X,A4,A3,5X,A4,1X,F8.0,3X,A4,2X,A4,2X,A4,1X,
     &    F8.0,3X,A4,2X,A4,F8.0,2X,F5.1,
     &    13X,F8.0,2X,F8.1)
      GO TO 3590
 3556 WRITE(6,3557)  SRUNAM(ITEM,1),SRUNAM(ITEM,2),SWUC(ITEM,1),
     &    SWUC(ITEM,2),LOD,LOS,COSTL,LOC,ISOD,COST,ISOB,ISOC,UCS(ITEM),
     &    FPCT,FMTBCT,TLCDF
 3557 FORMAT(' ',8X,2A4,2X,A4,A3,5X,A4,4X,A4,1X,F8.0,1X,A4,2X,A4,1X,
     &    F8.0,3X,A4,2X,A4,F8.0,3X,F4.0,
     &    13X,F8.0,2X,F8.1)
      GO TO 3590
C
 3558 WRITE(6,3559)  SRUNAM(ITEM,1),SRUNAM(ITEM,2),SWUC(ITEM,1),
     &    SWUC(ITEM,2),LOD,LOS,LOB,COSTL,ISOD,COST,ISOB,ISOC,UCS(ITEM),
     &    FPCT,FMTBCT,TLCDF
 3559 FORMAT(' ',8X,2A4,2X,A4,A3,5X,A4,4X,A4,2X,A4,1X,F7.0,2X,A4,1X,
     &    F8.0,3X,A4,2X,A4,F8.0,3X,F4.0,
     &    13X,F8.0,2X,F8.1)
      GO TO 3590
C
C      SRU IS BASE REPAIRED
 3570 COST=BASESC(ITEM)
      TOTSB=TOTSB+COST
      OPDECS(ITEM)=LOCAT(3)
      GO TO (3572,3574,3576,3578),LEVEL
```

```
3572 WRITE(6,3573)  SRUNAM(ITEM,1),SRUNAM(ITEM,2),SWUC(ITEM,1),
     &    SWUC(ITEM,2),COSTL,LOS,LOB,LOC,ISOD,ISOS,COST,ISOC,UCS(ITEM),
     &    FPCT,FMTBCT,TLCDF
3573 FORMAT(' ',8X,2A4,2X,A4,A3,2X,F8.0,3X,A4,4X,A4,2X,A4,2X,A4,4X,
     &    A4,1X,F8.0,'ERROR',F8.0,2X,F5.1,
     &    13X,F8.0,2X,F8.1)
      GO TO 3590
3574 WRITE(6,3575)  SRUNAM(ITEM,1),SRUNAM(ITEM,2),SWUC(ITEM,1),
     &    SWUC(ITEM,2),LOD,COSTL,LOB,LOC,ISOD,ISOS,COST,ISOC,UCS(ITEM),
     &    FPCT,FMTBCT,TLCDF
3575 FORMAT(' ',8X,2A4,2X,A4,A3,5X,A4,1X,F8.0,3X,A4,2X,A4,2X,A4,4X,
     &    A4,1X,F8.0,'ERROR',F8.0,2X,F5.1,
     &    13X,F8.0,2X,F8.1)
      GO TO 3590
3576 WRITE(6,3577)  SRUNAM(ITEM,1),SRUNAM(ITEM,2),SWUC(ITEM,1),
     &    SWUC(ITEM,2),LOD,LOS,COSTL,LOC,ISOD,ISOS,COST,ISOC,UCS(ITEM),
     &    FPCT,FMTBCT,TLCDF
3577 FORMAT(' ',8X,2A4,2X,A4,A3,5X,A4,4X,A4,1X,F8.0,1X,A4,2X,A4,4X,
     &    A4,1X,F8.0,1X,A4,F8.0,2X,F5.1,
     &    13X,F8.0,2X,F8.1)
      GOTO 3590
3578 WRITE(6,3579)  SRUNAM(ITEM,1),SRUNAM(ITEM,2),SWUC(ITEM,1),
     &    SWUC(ITEM,2),LOD,LOS,LOB,COSTL,ISOD,ISOS,COST,ISOC,UCS(ITEM),
     &    FPCT,FMTBCT,TLCDF
3579 FORMAT(' ',8X,2A4,2X,A4,A3,5X,A4,4X,A4,2X,A4,F7.0,2X,A4,4X,
     &    A4,1X,F8.0,'ERROR',F8.0,2X,F5.1,
     &    13X,F8.0,2X,F8.1)
      GO TO 3590
C
C     SRU IS CIRF REPAIRED
C
3680 COST=CIRFSC(ITEM)
      TOTSC=TOTSC+COST
      OPDECS(ITEM)=LOCAT(4)
      GO TO (3682,3684,3686,3688),LEVEL
3682 WRITE(6,3683)  SRUNAM(ITEM,1),SRUNAM(ITEM,2),SWUC(ITEM,1),
     &    SWUC(ITEM,2),COSTL,LOS,LOB,LOC,ISOD,ISOS,ISOB,COST,UCS(ITEM),
     &    FPCT,FMTBCT,TLCDF
3683 FORMAT(' ',8X,2A4,2X,A4,A3,2X,F8.0,3X,A4,4X,A4,2X,A4,2X,A4,4X,
     &    A4,1X,'ERROR',F8.0,F8.0,2X,F5.1,
     &    13X,F8.0,2X,F8.1)
      GO TO 3590
3684 WRITE(6,3685)  SRUNAM(ITEM,1),SRUNAM(ITEM,2),SWUC(ITEM,1),
     &    SWUC(ITEM,2),LOD,COSTL,LOB,LOC,ISOD,ISOS,ISOB,COST,UCS(ITEM),
     &    FPCT,FMTBCT,TLCDF
3685 FORMAT(' ',8X,2A4,2X,A4,A3,5X,A4,1X,F8.0,3X,A4,2X,A4,2X,A4,4X,
     &    A4,1X,'ERROR',F8.0,F8.0,2X,F5.1,
     &    13X,F8.0,2X,F8.1)
      GO TO 3590
3686 WRITE(6,3687)  SRUNAM(ITEM,1),SRUNAM(ITEM,2),SWUC(ITEM,1),
     &    SWUC(ITEM,2),LOD,LOS,COSTL,LOC,ISOD,ISOS,ISOB,COST,UCS(ITEM),
     &    FPCT,FMTBCT,TLCDF
3687 FORMAT(' ',8X,2A4,2X,A4,A3,5X,A4,4X,A4,1X,F8.0,1X,A4,2X,A4,4X,
```

```
     &    A4,2X,A4,F7.0,1X,F8.0,2X,F5.1,
     &    13X,F8.0,2X,F8.1)
      GOTO 3590
 3688 WRITE(6,3689)  SRUNAM(ITEM,1),SRUNAM(ITEM,2),SWUC(ITEM,1),
     &    SWUC(ITEM,2),LOD,LOS,LOB,COSTL,ISOD,ISOS,ISOB,COST,UCS(ITEM),
     &    FPCT,FMTBCT,TLCDF
 3689 FORMAT(' ',8X,2A4,2X,A4,A3,5X,A4,4X,A4,2X,A4,F7.0,2X,A4,4X,
     &    A4,1X,'ERROR',F8.0,F8.0,2X,F5.1,
     &    13X,F8.0,2X,F8.1)
C
 3590 LINES=LINES+1
 3600 CONTINUE
C
C      WRITE COST TOTALS
      WRITE(6,3604)
 3604 FORMAT(' ',39X,'----------  LC  $/BASE  ----------')
      WRITE(6,3605)
 3605 FORMAT('0',35X,7X,'DEPOT',7X,'SCRAP',8X,'BASE',7X,'CIRF',14X,
     &    'WHOLESALE CHANGE FACTORS')
      WRITE(6,3610) TOTLD,TOTLS,TOTLB,TOTLC,WCFUC
 3610 FORMAT(' TOTAL LRU COSTS',20X,4F12.0,19X,'COST =',F6.2)
      WRITE(6,3620) TOTSD,TOTSS,TOTSB,TOTSC,WCFF
 3620 FORMAT(' TOTAL SRU COSTS',20X,4F12.0,19X,'MTBF =',F6.2)
      WRITE(6,3630)  TOTSED,TOTSEB,TOTSEC,WCFSE
 3630 FORMAT(' ','TOTAL SE COSTS',21X,F12.0,12X,F12.0,12X,F12.0,
     & 9X,'SE =',F6.2)
      DEV=DEV/M
      WRITE(6,3635) DEV
 3635 FORMAT(' ','SE DEVELOPMENT COST',24X,F12.0)
      GTOT=TOTLD+TOTLS+TOTLB+TOTSD+TOTSS+TOTSB+TOTSED+TOTSEB+DEV
     & +TOTLC+TOTSEC+TOTSC
      WRITE(6,3640)  GTOT
 3640 FORMAT('0',5X,'TOTAL COST OF REPAIR LEVEL DECISIONS =',
     &    F12.0/20('*'))
      RETURN
      END
```

Appendix H: SECMP Subroutine

```
      SUBROUTINE SECMP(IND,MSGL1)
C
C
      IF(IND.EQ.0)  GO TO 806
      IPAGE=IPAGE+1
  800 WRITE(6,801)  (SYSNAM(ISUB),ISUB=1,3),DATE,IPAGE
  801 FORMAT('1',2X,3A4,86X,A8,5X,'PAGE',I4)
      WRITE(6,802)  CHARS(2),CHARS(2)
  802 FORMAT(' ',43X,'COMPUTED SE COSTS',2X,2A4)
       WRITE(6,803)  PIUP
  803 FORMAT('0',4X,'SE',9X,'SE',9X,'*** NO. ***   *** SE USE ***'/
     &    ' ',3X,'CODE',7X,'NAME',8X,'CURR   NEW',8X,'HRS/MO ',4X,
     &    '%USE',15X,F4.0,' YR',14X,'TOTAL SE')
       WRITE(6,804)
  804 FORMAT(' ',26X,'INV    REQD     CURR    NEW',12X,'ACQ. $',4X,
     &    'OPER. $',5X,'FAC. $',2X,'LC $/BASE')
       WRITE(6,805)
  805 FORMAT(' DEPOT')
       LINES=7
       IF(NDSE.GT.0) GO TO 806
       WRITE(6,8050)
 8050 FORMAT(' BASE')
       LINES=LINES+1
       IF (NBSE.GT.0) GOTO 806
       WRITE(6,8051)
 8051 FORMAT(' CIRF')
C
  806 IF(NUMSER.LT.1)  RETURN
       DO 900 I900=1,NUMSER
       SAVCST=SECOST(I900)
C        COMPUTE TOTAL USAGE HOURS FOR THE SE
       ITYPE=SECODE(I900)/10000
       JTYPE=(SECODE(I900)/1000)-(ITYPE*10)
       TMHRS=USEHRS(I900)
       TMRQT=REQMT(I900)
       USEHRS(I900)=0.
       NEXT=SEPF(I900)
       IF(NEXT.EQ.0)  GO TO 815
       LAST=SEPL(I900)
  807  ITMPTR=SEXREF(NEXT)
       IF(ITMPTR.GT.0.AND.OPDECL(ITMPTR).NE.LOCAT(15)) GOTO 811
       IF(ITMPTR.LT.0.AND.OPDECS(0-ITMPTR).NE.LOCAT(15)) GOTO 811
       IF(ITYPE.GT.4.AND.ITYPE.LT.9)  GO TO 809
       IF(ITYPE.EQ.9) GOTO 8090
       IF(ITMPTR.LT.0)  GO TO 808
       USEHRS(I900)=USEHRS(I900)+FSEUHD(ITMPTR)
       GO TO 811
  808  ITMPTR=0-ITMPTR
       USEHRS(I900)=USEHRS(I900)+SSEUHD(ITMPTR)
       GO TO 811
  809 IF(ITMPTR.LT.0)  GO TO 810
```

```
      USEHRS(I900)=USEHRS(I900)+FSEUHB(ITMPTR)
      GO TO 811
 8090 IF(ITMPTR.LT.0) GOTO 8095
      USEHRS(I900)=USEHRS(I900)+FSEUHC(ITMPTR)
      GOTO 811
 8095 ITMPTR=0-ITMPTR
      USEHRS(I900)=USEHRS(I900)+SSEUHC(ITMPTR)
      GOTO 811
  810 ITMPTR=0-ITMPTR
      USEHRS(I900)=USEHRS(I900)+SSEUHB(ITMPTR)
  811 IF(NEXT.EQ.LAST)  GO TO 814
      NEXT=NXTITM(NEXT)
      IF(NEXT.GT.0)  GO TO 807
  812 WRITE(6,813) I900,SEPF(I900),SEPL(I900),NEXT,SEXREF(NEXT),
     &          NXTITM(NEXT)
  813 FORMAT('OTROUBLE IN SEXREF',6I6)
      STOP
  814 IF(NXTITM(NEXT).EQ.0)  GO TO 815
      WRITE(6,8140)
 8140 FORMAT(' NEXT = 0')
      GO TO 812
C      COMPUTE # OF UNITS OF THE SE REQUIRED
  815 IF(ITYPE.EQ.1.OR.ITYPE.EQ.5.OR.JTYPE.EQ.1)  GO TO 817
C           FOR PECULIAR SE
      IF (JTYPE.GT.1) GOTO 816
      IF (USEHRS(I900).GT.HRAVSE(I900)) GOTO 818
      REQMT(I900)=0
      GOTO 819
  816 TOTHRS=TMHRS+USEHRS(I900)
      REQMT(I900)= AINT(2.+((TOTHRS*OS)/OPHRS(I900)))+
     & AINT((TOTHRS*(1.-OS))/OPHRS(I900))
      IF(TMROT.GT.0) REQMT(I900)=REQMT(I900)-TMROT
      GOTO 819
  818 REQMT(I900)= AINT(1+((USEHRS(I900)-HRAVSE(I900))/OPHRS(I900)))
  819 SEUR(I900)=(USEHRS(I900)+TMHRS)/((REQMT(I900)+TMROT)*OPHRS(I900))
      IF(ITYPE.NE.4.AND.ITYPE.NE.8.AND.JTYPE.NE.4)  GO TO 840
      REQMT(I900)=1
      TMROT=0
      SEUR(I900)=0.
      GO TO 840
C           FOR COMMON SE
  817 REQMT(I900)=0.
      IF (JTYPE.EQ.1) GOTO 821
      AVHRS=(OPHRS(I900)-BSYHRS(I900))*NSECI(I900)-TMHRS
  820 IF(AVHRS.GT.USEHRS(I900))  GO TO 830
      REQMT(I900)=REQMT(I900)+1.
      AVHRS=AVHRS+OPHRS(I900)
      GO TO 820
  821 TOTHRS=TMHRS+USEHRS(I900)+BSYHRS(I900)
       REQMT(I900)=2.+AINT((TOTHRS*OS)/OPHRS(I900))+
     & AINT((TOTHRS*(1.-OS))/OPHRS(I900))
       REQMT(I900)=REQMT(I900)-TMROT-NSECI(I900)
```

```
      GOTO 830
C
  830 SEUR(I900)=(USEHRS(I900)+NSECI(I900)*BSYHRS(I900)+TMHRS)/
     &      (NSECI(I900)*OPHRS(I900)+(REQMT(I900)+TMRQT)*OPHRS(I900))
C
C      COMPUTE SE ACQUISITION & OPERATIONS COST
  840 SEACQ=REQMT(I900)*CADB(I900)
      SEOPNS=USEHRS(I900)/OPHRS(I900)*CODB(I900)*PIUP
      SEFAC=FDB(I900)
      IF(TMRQT.GT.0) SEFAC=0
      SECOST(I900)=SEACQ+SEOPNS+SEFAC
      IF(ITYPE.GT.4.AND.ITYPE.LT.8)  GO TO 850
      IF (ITYPE.EQ.9.AND.JTYPE.EQ.4) GOTO 850
      SECOST(I900)=SECOST(I900)/M
C      SAVE TOTAL COST FOR THE NETWORK SE COST ARC
  850 SECOST(I900)=AINT(SECOST(I900)+0.5)
      IF(IND.EQ.0)  GO TO 880
      IF(LINES.LT.55)  GO TO 855
      IPAGE=IPAGE+1
      WRITE(6,801)  (SYSNAM(ISUB),ISUB=1,3),DATE,IPAGE
      WRITE(6,802)  CHARS(5),CHARS(6)
      WRITE(6,803)  PIUP
      WRITE(6,804)
      LINES=7
  855 CHOURS=0.
      IF(ITYPE.EQ.1.OR.ITYPE.EQ.5.OR.JTYPE.EQ.1)
     & CHOURS=NSECI(I900)*BSYHRS(I900)
      REQMT(I900)=REQMT(I900)+TMRQT
      USEHRS(I900)=USEHRS(I900)+TMHRS
      SEACQ=REQMT(I900)*CADB(I900)
      SEOPNS=(USEHRS(I900)/OPHRS(I900))*CODB(I900)*PIUP
      SEFAC=FDB(I900)
      TSECST(I900)=TSECST(I900)+SECOST(I900)
      TSECST(I900)=AINT(TSECST(I900)+0.5)

      WRITE(6,860) SECODE(I900),(XSE(I900,ISUB),ISUB=1,3),NSECI(I900),
     &       REQMT(I900),CHOURS,USEHRS(I900),SEUR(I900),SEACQ,SEOPNS,
     &       SEFAC,TSECST(I900)
  860 FORMAT(' ',3X,I5,3X,3A4,3X,I4,4X,I4,2X,2F7.1,2PF6.1,0P4F11.0)
      LINES=LINES+1
      IF(I900.LE.NDSE)  GO TO 880
      IF(I900.GT.NDSE) GOTO 875
      WRITE(6,870)
  870 FORMAT(' BASE')
      LINES=LINES+1
      GOTO 880
  875 IF(I900.NE.NDSE+NBSE) GOTO 880
      WRITE(6,877)
  877 FORMAT(' CIRF')
      LINES=LINES+1
C
```

```
880 IF(IND.EQ.1)  GO TO 900
    SEPTR=SEARCP(I900)
    CAP(SEPTR)=SECOST(I900)
    SECOST(I900)=SAVCST
900 CONTINUE
    DO 1 I=1,NUMSER
    REQMT(25)=REQMT(25)+REQMT(I)
1   CONTINUE
    RETURN
    END
```

*

Bibliography

1. Aerospace Studies Institute. Evaluation of Research and Development and Procurement Support of Operations in Southeast Asia (1 Jan 1965 - 31 Mar 1968)(U), Corona Harvest Report, Air University, Maxwell AFB, Alabama, Dec 1970.

2. AFALC/XRS. Network Repair Level Analysis Model Programmers Guide. Wright-Patterson AFB, Ohio. HQ AFLC, Nov 1981.

3. Armstrong, Lt Col Gerald R. Unpublished notes on "Max-flow Labeling Algorithm," Air Force Institute of Technology, Wright-Patterson AFB, Ohio, September 1982.

4. Barnes, J. Wesley and Paul A. Jensen. Network Flow Programming, New York: John Wiley and Sons Inc., 1980.

5. Bazaraa, Mokhar S. and John J. Jarvis. Linear Programming and Network Flows, New York: John Wiley and Sons Inc., 1977.

6. Bodnar, Al and James Callahan, Operations Research Analysts SDBE/B1-B SPO, ASD/AFSC. Personal Interview. Wright-Patterson AFB, Ohio, 6 July 1983.

7. Bradley, Gordon H., Gerald G. Brown, and Glenn W. Graves. "Design and Implementation of Large Scale Primal Transshipment Algorithms", Management Science, Vol.24, No.1, 1-34, September 1977.

8. Briskin, Larry and Terry Mitchell, Network Repair Level Analysis Models User's Guide, AFALD/XRS(AFLC), Wright-Patterson AFB, Ohio.

9. Briskin, Lawerence and Terry Mitchell, Operations Research Analysts AFALD/XRS, HQ AFLC. Personal Interview. Wright-Patterson AFB, Ohio, 27 June 1983.

10. Clark, Lt Col Thomas. "Policy Analysis Model for the Air Force Logistics System Report #1. Air Force Institute of Technology, Wright-Patterson AFB, Ohio,

11. Dept. of the Air Force. Integrated Logistic Support (ILS) Program. AFR 800-8. Wright-Patterson AFB, Ohio. HQ AFLC.

12. Dept. of the Air Force. Optimum Repair Level Analysis. AFLC Pamphlet 800-4. Wright-Patterson AFB, Ohio. HQ AFLC/AFSC, Oct 1970.

13. Dept. of the Air Force. Repair Level Analysis Procedures, AFLC/AFSC Pamphlet 800-4 (Final Draft). Wright-Patterson AFB, Ohio. HQ AFLC/AFSC, June 1983.

14. Dept. of the Air Force. Repair Level Analysis (RLA) Program. AFR 800-28. Wright-Patterson AFB, Ohio. HQ AFSC/AFLC, 29 May 1981.

15. Dessouky, Mohamed I. and Steve Phillips, Jr., "The Cut Search Algorithm With Arc Capacities and Lower Bounds", Management Science, Vol.25, No.4, 396-404 (April 1979).

16. Edmonds, Jack, and Richard M. Karp, "Theoretical Improvements in Algorithmic Efficiency for Network Flow Problems", Journal of the Association for Computing Machinery Vol. 19, No. 2, 248-264 (April 1972).

17. Ford, L. R., Jr. and Fulkerson, D. R., Flows in Networks, Princeton, New Jersey: Princeton University Press, 1962.

18. Glover, F., J. Hultz, D. Klingman, and J. Stutz, "Generalized Networks: A Fundamental Computer-Based Planning Tool", Management Science, Vol.24, No.12, 1209-1220 (August 1978).

19. Horowitz, Ellis, and Sartaj Sahni, Fundamentals of Computer Algorithms. Potomac, Maryland: Computer Science Press, Inc., 1978.

20. James, J. H., and L. D. Servi,"The Use of Network Theory to Solve an ORLA Problem", Working Paper No. 21418. Bedford, Mass: The MITRE Corporation, Bedford Operations, September 1977.

21. Rhys, J. M. W., "A Selection Problem of Shared Fixed Costs and Network Flows", Management Science, Vol.17, No.3: 200-207 (November 1970).

22. Trempe, Maj Robert E., USAF, Squadron Leader Herbert E.Trichlin RAAF, and Lt Col Thomas Clark, "Complex Multi-Echelon Inventory System Management Using A Dynamic Simulation Model",Decision Sciences, Vol. 14, 389-407 (July 1983)

VITA

Captain Gary W. Arnett was born on 1 April, 1952 in Portsmouth, Virginia. He is married to the former Michelle Rodgers of Arlington, Virginia. They have two children, Becky and Brian, ages 4 and 1 1/2 respectively. He received a B.S. in Industrial Engineering and Operations Research from VPI in 1974. As a US Army Engineer Officer, he has served as a Combat Engineer Platoon Leader in the Federal Republic of Germany and as Chief of the Engineering and Management Analysis Branch of the Facility Engineer at Fort Belvoir, VA. Since his transfer to the US Air Force in November of 1980, he has served as an O.R. analyst in the ASD/AFLC Joint LCC Working Group at Wright-Patterson AFB until his acceptance at the Air Force Institute of Technology.

Permanent Address: 4713 Red Coat Road
Va. Beach, VA 23455

Captain Nicholas W. Reybrock was born on 16 July, 1948 in Plainville, Massachusetts. He is married to the former Kathleen Ann Marko of Cheyenne, Wyoming. They have one daughter, Kasey age twelve. He attended the University of Utah under the Airman Education Commissioning Program and graduated Magna cum Laude in 1976, with a B.S. in Chemistry. Captain Reybrock has 16 years in the Air Force. During his first 9 years he served as an electronics technician and as a chemistry laboratory technician at McClellan AFB CA, and as a gas chromatography technician at Eilison AFB AK. After receiving his comission in 1976, he has served as a Weapons Controller at a mobil TAC unit and a remote Alaskan radar site. He has also served as a Air Surveillance Officer and Radar Inputs and Countermeasures Officer at the 21st NORAD Region in New York. He was serving as a Operations Staff Officer and Standardization and Evaluation Officer at the 21st NORAD Region prior to his accepatance at the Air Force Institute of Technology.

Permanent Address : 106 No. Bedford Street
Georgetown DE 19947

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution unlimited
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) AFIT/GOR/OS/83D-3	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION School of Engineering 6b. OFFICE SYMBOL (If applicable) AFIT/EN	7a. NAME OF MONITORING ORGANIZATION
6c. ADDRESS (City, State and ZIP Code) Air Force Institute of Technology Wright-Patterson AFB, Ohio 45433	7b. ADDRESS (City, State and ZIP Code)
8a. NAME OF FUNDING/SPONSORING ORGANIZATION Concepts and Analysis Div AFALC 8b. OFFICE SYMBOL (If applicable) XRS/AFALC	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (City, State and ZIP Code) Wright-Patterson AFB, Ohio 45433	10. SOURCE OF FUNDING NOS. PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT NO.
11. TITLE (Include Security Classification) see box 19	

12. PERSONAL AUTHOR(S) Gary W. Arnett Captain, USAF
Nicholas W. Reybrock Captain, USAF

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Yr., Mo., Day)	15. PAGE COUNT
MS Thesis	FROM ______ TO ______	December 1983	151

16. SUPPLEMENTARY NOTATION

Approved for public release: IAW AFR 190-17.
LYNN E. WOLAVER 7 May 84
Dean for Research and Professional Development
Air Force Institute of Technology (ATC)
Wright-Patterson AFB OH 45433

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB. GR.	

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

Title : ENHANCEMENTS TO THE NETWORK REPAIR LEVEL ANALYSIS (NRLA) MODEL USING MARGINAL ANALYSIS TECHNIQUES AND CENTRALIZED INTERMEDIATE REPAIR FACILITY (CIRF) MAINTENANCE CONCEPTS

Thesis Chairman : Gerald R. Armstrong, LtCol, USAF

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT. ☐ DTIC USERS ☐	21. ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED
22a. NAME OF RESPONSIBLE INDIVIDUAL Gerald R. Armstrong, LtCol, USAF	22b. TELEPHONE NUMBER (Include Area Code) 22c. OFFICE SYMBOL

DD FORM 1473, 83 APR   EDITION OF 1 JAN 73 IS OBSOLETE.   UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

Applied network theory and marginal analysis concepts were utilized to design, computerize, verify, and evaluate, three major software modifications to the Network Repair Level Analysis (NRLA) model. First, a preprocessor subroutine using marginal analysis techniques was developed and tested to reduce the computer processing requirements of the program. Second, a new network labeling algorithm which solve the max-flow min-cut problem is presented. This algorithm performs 100 times faster than the current algorithm and 73 times faster than the highly efficient, commercially available, primal networking code known as GNET. Third, for the first time a networking structure has been designed which allows for the inclusion of Centralized Intermediate Repair Facilities (CIRF) in the repair level analysis decision process.

These products greatly expand the NRLA model's capability while at the same time improving its operational efficiency. Through their integration and use, System Program Managers have a comprehensive analytical tool to effectively conduct repair level analysis and to design more cost-effective logisitical structures to support the operation of Air Force systems.

The NRLA program is hosted on the CREATE Operating System and contains approxiamately 5500 lines of computer code. It consists of a main routine and twelve major subroutines. The results from the NRLA model are used by logistical planners to quantify the potential cost impacts associated with alternative maintenance plans. As the technological complexity of weapons systems has increased new and innovative logisitcal support systems are required to maximize the system's operational capability while minimizing life cycle costs. The above enhancements to the NRLA model were designed to meet these new challenges. This research effort was sponsored by the Concepts and Anaylsis Division, Air Force Acquisition Logistics Center (XRS/AFALC/AFLC).

END

FILMED

DTIC