ANALYSIS AND SOLUTION ALGORITHMS OF SEALIFT ROUTING AND SCHEDULING PROBLEMS: FINAL REPORT

Harilaos N. Psaraftis
James B. Orlin
Daniel Bienstock
Paul M. Thompson

Sloan W.P. No. 1700-85 August 1985

ABSTRACT

This report describes a computer-assisted methodology for solving the Operational Routing and Scheduling Problem of the Military Sealift Command. This is the problem of assigning cargoes to available ships in an emergency situation so that (a) cargoes reach their destinations within prescribed time limits, (b) some prescribed system performance measure is optimized and (c) other constraints are satisfied. The methodology is well-suited to an (eventual) "expert system" approach to solving hard combinatorial problems. Its principles are most appropriate for large-scale dynamic allocation problems. Thus, the approach is modular, hierarchical, interactive, adaptive and evolutionary. The computer program implementing this methodology has been specifically designed to handle the special complexities of this problem, while at the same time allowing for future extensions and enhancements. It decomposes the overall problem by time and uses a network flow algorithms to optimally solve subproblems in conjunction with a utility function that is a function of ship utilization, timely delivery and cargoes, and port congestion.

This report critically assesses the structure and complexity of this class of problems, describes the proposed methodology in detail, along with an evaluation of its advantages over alternative approaches, presents computational experience with the procedure and outlines current and future plans of this project.

FOREWORD AND ACKNOWLEDGEMENTS

This work was performed at MIT under the supervision of the first two authors (who have been the co-principal investigators of the project), and under the assistance of the other two authors (who have been principal research assistants on the project). The work has been supported in part by Contract No. N00016-83-K-0220 of the Office of Naval Research and by an internal grant of the MIT Center for Transportation Studies. This report covers activities on this project up to May, 1985.

The authors would like to acknowledge the contributions of several individuals who assisted the project in various ways. These include graduate students Kung-Yeun Jeng, Tai-Up Kim, and Vijay Singhal, who worked as research assistants on early phases of the project, as well as undergraduates Tom Berger and Charles Marge who volunteered their help during the 1984-85 academic year. We would also like to thank Joe Ballou, Kathy Wiswesser, Bob Elwell and (especially) Jon Kaskin of the Military Sealift Command for their valuable technical assistance, advice and feedback on our work. We gratefully acknowledge the administrative support of Neal Glassman of ONR, Marcia Ryder of the MIT Operations Research Center and Pat McCosco of the MIT Center for Transportation Studies. We are grateful to Bob Simpson, Dennis Matthaisel, Boon-Chai Lee and Jang-Jei Jaw of the MIT Flight Transportation Laboratory for the input they provided into this project as it relates to routing and scheduling for the Military Airlift Command. Last, but not least, we would like to thank Karuna Mohindra for the excellent work she has done in typing the manuscript.

The opinions and views expressed in this report are solely those of the authors and do not necessarily represent the position of the Military Sealift Command.

TABLE OF CONTENTS

CHAPTER 1

EXECUTIVE SUMMARY

1.1 Introduction: Project Objectives

The purpose of this report is to describe research carried out at MIT in conjunction with the project "Analysis and Solution Algorithms of Sealift Routing and Scheduling Problems" (referred to from now on as "the MIT Sealift Routing and Scheduling Project", or as "the Sealift Project", or simply as "the project"). This report covers activities on the project up to May, 1985.

Progress in the Sealift Project to date has been in accordance with both the scientific objectives as set forth in the project's original proposal and with the schedule that was proposed to achieve these objectives (Psaraftis and Orlin, 1982). Specifically, on a short-term basis we had proposed to (a) investigate a class of sealift routing and scheduling problems, (b) develop, analyze and test solution algorithms for such problems and (c) work with the MSC and others so as to increase our knowledge of these problems. On a longer-term basis, we had set forth the following goals: (d) develop a procedure that could be ultimately implemented within the scheduling subsystem of the SEASTRAT system of the Military Sealift Command (MSC), (e) enhance the state of the art in the solution of large-scale scheduling, distribution, and transportation problems, and (f) advance the state of knowledge in interactive user-friendly algorithms.

The principal product for our work thus far is what has been named

the MIT Ocean Routing and Scheduling System, or MORSS. MORSS is a computer program that solves the Operational Routing and Scheduling Problem of the MSC, that is, the problem of assigning cargoes to available ships in an emergency situation so that (a) cargoes reach their destinations within prescribed time limits, (b) some prescribed system performance measure is optimized and (c) other constraints are satisfied (details follow). As will be seen shortly, MORSS possesses features that have been specifically designed to allow the handling of the special complexities of this problem, while at the same time allowing for future extensions and enhancements.

The rest of this chapter summarizes progress in the project to date, whereas details of the work appear in the remaining chapters. We begin by assessing the structure and complexity of this class of problems, an assessment which is critical as to which approach is more suitable for such problems. We go on to describe the MORSS general methodology, along with an evaluation of its advantages over alternative approaches. This chapter also briefly describes computational experience with the MORSS algorithm, miscellaneous other accomplishments of the project to date, and summarizes current and future plans.

1.2 Problem Structure and Complexity

An early attempt was first made by the MIT team to determine the overall degree of complexity of the MSC Operational Routing and Scheduling Problem, by relating its structure to other routing and scheduling problems that have been tackled in the past. A timely assessment on this score was considered important in order to guide subsequent algorithmic design efforts. Specifically, we wished to assess early on in the project whether

there was potential for either exact (optimization) approaches or optimization-based heuristics for the solution of this problem. We should note here that before this project was initiated we had examined exact, mathematical programming formulations of several variants of the general MSC problem (Bardjis, 1982). This work examined several alternative objective functions, constraints, etc. Nevertheless, no further attempt was made to develop solution approaches for those formulations, mainly because of their complexity.

After a fair amount of analysis, we concluded that pursuing exact approaches for this problem was not a particularly promising direction. Such an assessment can be justified on several grounds. For instance, it is clear that a special (and quite restrictive) version of the MSC problem is identical to the multi-vehicle many-to-many advance-request dial-a-ride problem. The latter is the problem of carrying customers from distinct origins to distinct destinations within specified pickup and delivery time windows. Such problems can be considered as a special case of the MSC problem, the case in which all ships and cargoes are assumed identical and no ship/cargo/port restrictions exist. The difficulty of the dial-a-ride problem can be understood by the fact that about 15 years of research and algorithm development for this problem have resulted in essentially only one class of exact algorithms, developed for the single-vehicle problem and viable only for very small-sized problems (Psaraftis, 1980, 1983a).

There have also been several multi-vehicle dial-a-ride algorithms developed, but all of them are heuristic. These include the work of groups at the University of Maryland (Bodin and Sexton, 1982), Georgia Tech (Jarvis et al, 1980), MIT (Jaw et al, 1982, 1984; Jaw, 1984) and others.

We investigated the potential of "transferring" to the MSC problem the methodologies developed by these dial-a-ride projects. In particular, Jeng (1984) investigated the potential of developing a sequential insertion algorithms for the MSC using the technique developed for dial-a-ride in Jaw et al, (1984), but the resulting algorithm was never implemented. We finally decided against such adaptations, mainly because of the substantial algorithm redesign that such extensions would involve.

Another problem that resembles a special case of the MSC problem is one analyzed extensively by Fisher et al (1982) for the routing of a fleet of trucks in the chemical industry. Their method is based on a heuristic that generates routes, a formulation as a set-packing problem and a solution using Lagrangian relaxation. Our original project proposal (Psaraftis and Orlin, 1982) had suggested that we investigate the suitability of such an approach for the MSC problem. We eventually decided not to pursue this approach in our project for several reasons. First, to avoid duplication of effort with the peacetime tanker scheduling project being carried out by Fisher and his team at the University of Pennsylvania (project also sponsored by ONR). Second, because we thought that adapting such an approach to the MSC operational problem (particularly to the breakbulk component of it) would require (at a minimum) a nontrivial amount of redesign, most of it heuristic, to allow the scheduler to handle the special features of the problem, that is, dynamic updates of input data, rolling horizon, nonhomogeneous cargoes, etc. In that respect, we decided to go ahead with a procedure that would be, by design, tailored to the dynamic nature of the MSC operational problem.

We also reviewed the work of Science Applications Inc (SAI, 1982)

in terms of suitability for the MSC operational problem. SAI developed a heuristic algorithm for the "deliberate planning" version of the MSC problem ("Scheduling Algorithm for Improving Lift", or SAIL). We have concluded that the SAI algorithm would have to undergo extensive structural modifications to be able to function in an operational setting (see also Chapter 2). The main reasons for such a conclusion can be traced to the anticipated difficulty in providing that procedure with the "restart" capability and the appropriate data structure design that are necessary for the MSC operational problem.

1.3 Solution Approach: The MORSS Algorithm

The MSC operational problem is by its nature too complex to be solved by machine alone. As such it requires expert judgement as to how to assess tradeoffs and how to schedule appropriately. Our fundamental methodology is well suited to an (eventual) "expert system" approach to solving hard combinatorial problems. Thus, our focus is on "principles of heuristic design" rather than on optimization techniques for special classes of problems.

The principles that we have focused on are most appropriate for dynamic allocation problems. As we have already discussed in a previous progress report (Orlin and Psaraftis, 1983a) our approach is modular, hierarchical, interactive, adaptive and evolutionary. These generic features - which we consider essential for any algorithm that is developed for the MSC operational problem - are also discussed in Chapter 3 of this report. As a result of these heuristic design principles and many other considerations, we developed the MIT Ocean Routing and Scheduling System

(MORSS), a general overview of which goes as follows:

MORSS operates on a "rolling horizon" basis. Referring to Figure 1.1, let T be the total duration of the overall scheduling process for the problem at hand (that is, from the beginning of the process until all cargoes have been delivered). Let also a and L be user inputs such that $0 \leq a \leq 1$ and $0 < L \leq T$.

MORSS decomposes the overall problem by time, using these parameters. At each iteration it only assigns (to ships) cargoes whose EPT's (earliest pickup times) fall within a "time window" of $(t_k, t_k + L)$ where t_k is the beginning of the window at the (current) k^{th} iteration and L the length of the current time window. Assignments at the k^{th} iteration are initially made on a tentative basis, by taking into account, among other things, (permanent) assignments already committed to at previous iterations.

Once tentative assignments at the k^{th} iteration are made, only those assigned cargoes whose EPT's are between t_k and $t_k + aL$ are eligible for "permanent assignment", which is granted if they meet some additional criteria (more details on the assignment/deassignment procedure will be given shortly). All other cargoes, that is, all cargoes that have not been assigned, plus all tentatively assigned cargoes whose EPT's are beyond $t_k + aL$, return to the pool of unassigned cargoes and are examined at a future iteration. Once assignments at iteration k become permanent, the time horizon is "rolled" to the interval $(t_{k+1}, t_{k+1} + L)$, with t_{k+1} being the earliest EPT of all yet unassigned cargoes. Note that t_{k+1} need not be smaller than $t_k + aL$, but will not be greater than $t_k + L$ (see Figure 1.1), barring the circumstance of no cargoes having EPT's between t_{k+1} and $t_k + L$.

ROLLING HORIZON

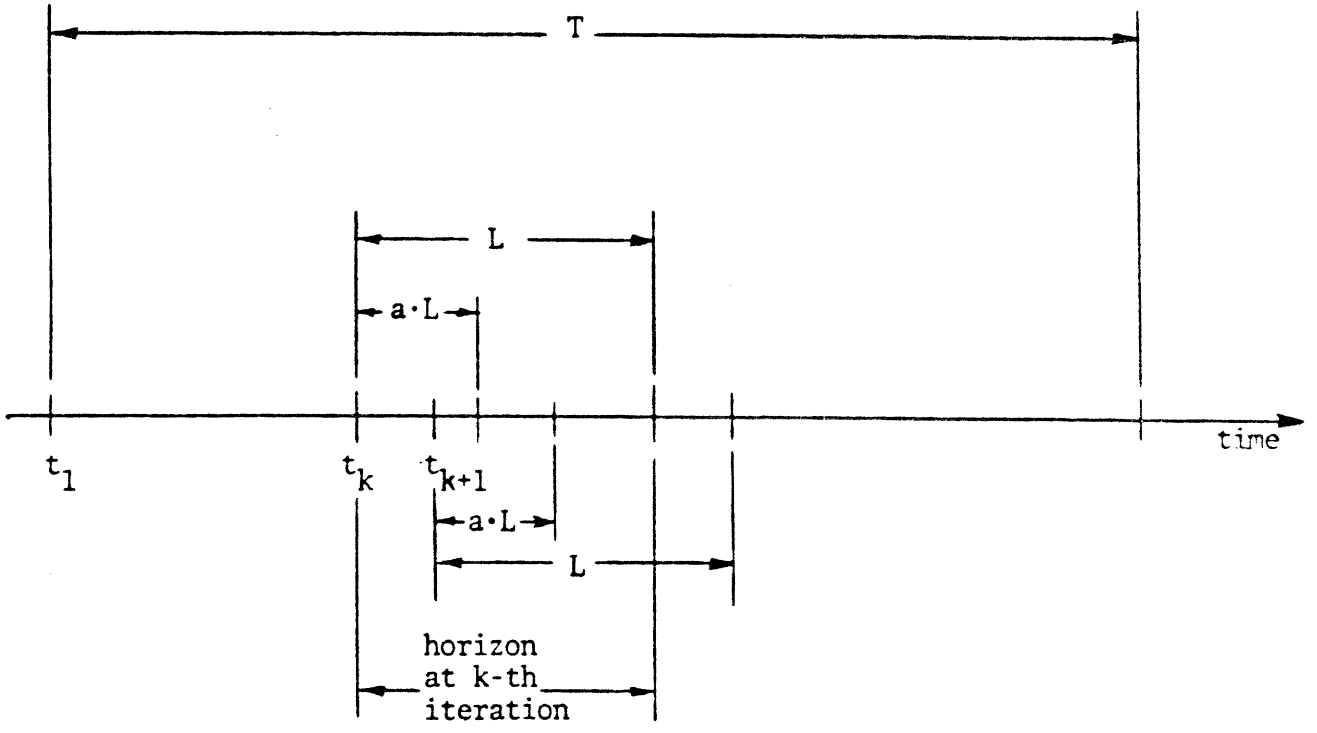


Figure 1.1

One can see that in such a fashion there is always an overlap between two consecutive iterations, and that by considering all cargoes within a time window of length L (instead of a.L), MORSS has a "look ahead" capability. In that way, permanent assignments at the current iteration take into account information on cargoes that will be handled in the future. The values of a and L depend on T, the cargo demand rate and other inputs, and are subject to calibration. For a typical duration of T = 180 days, reasonable values for a and L might be a = 0.5 and L = 14 days, in which case MORSS would "look" at two weeks of data at a time but would make permanent assignments only one week at a time.

Another concept central to MORSS is the concept of "seed" assignments. This is a one-to-one assignment of some cargoes (seed cargoes) to some ships (seed ships) early on in the scheduling process so that a good starting solution is obtained. Such a solution serves as a "skeleton" for the final schedule, which gradually evolves from the seed schedule as subsequent assignments are made at future iterations. Seed selection is performed by solving an assignment problem whose objective function is the maximization of the total "utility" of the assignment. For each eligible seed cargo/ship pair, a special subroutine is called to compute the "utility" of the corresponding pair (more on the utility function below). The assignment problem is solved by using a novel network flow algorithm developed by Orlin (1983a, 1983b) for solving the transportation problem.

The same network flow algorithm is used each time a subsequent assignment is made, that is, for every time window of the rolling horizon, and until the end of the horizon. The main differences between seed

assignments and subsequent assignments are (a) seed assignments are made on a one-to-one basis whereas in subsequent assignments more than one cargo can be simultaneously assigned to a ship, and (b) subsequent assignments take into account assignments already made at prior iterations by computing not only the utility of a particular ship/cargo pair, but also the change in utility in all previous cargo assignments due to the addition of a new cargo on a ship.

Adding a new cargo on a ship is done by inserting the origin and destination of the cargo into the current schedule of the ship. To avoid excessive computations, a simple heuristic determines a good pickup insertion. Given this insertion, the total change in utility of the schedule of the ship is computed for each feasible delivery insertion by considering the following components:

(a) The change in total "delivery time utility" of all cargoes on the ship. The delivery time utility of each cargo is assumed to be a "bell shaped" decreasing function of the tardiness of that cargo (the tardiness being defined as the delay of the cargo beyond its latest delivery time if the delay is positive, and zero otherwise).

(b) The change in "utility due to ship utilization and schedule flexibility". For each ship, such utility is assumed to be a two dimensional "bell-shaped" decreasing function of the ratio of the ship's residual capacity over its nominal capacity and an increasing function of the total allowable slack in the ship's schedule.

(c) The change in "port congestion utility" due to both pickup and delivery. For each port, such utility is assumed to be a decreasing function of the port's utilization, defined as the ratio of ships visiting

the port over the capacity of the port, in ships.

The final choice for delivery insertion is the one that yields the maximum overall utility.

Figure 1.2 exhibits the structure of MORSS. Only major modules of the procedure are shown and "blown-up". Minor routines, which perform arithmetic calculations, keep track of schedules and perform other database management functions, are not shown. Also not shown is the way the data structures of this process are organized and linked together. These have been organized in a very robust cross-reference system, using list-processing techniques. We believe that these sophisticated data structures are an integral part of our contribution to a viable solution methodology for the operational problem and serve a function which is crucial for the ability of the scheduler to tackle the problem efficiently (see Appendix A for more details). The code has been written in PASCAL, which is superior to FORTRAN (and certainly COBOL) with respect to ease of programming and handling of data structures. Further details on the MORSS computer program can be found in Appendices B and C.

MORSS cannot be directly compared with procedures such as SEACOP or the SAI algorithm, which have been developed for the deliberate planning (rather than the operational) MSC problem. Nevertheless, we conclude this subsection with a summary of ways by which we feel MORSS has enhanced the state of knowledge in solution tools for this specific class of problems and in heuristic design for complex problems in general:

(a) MORSS provides a measure of overall system performance. The actual utility achieved can be measured against the maximum possible

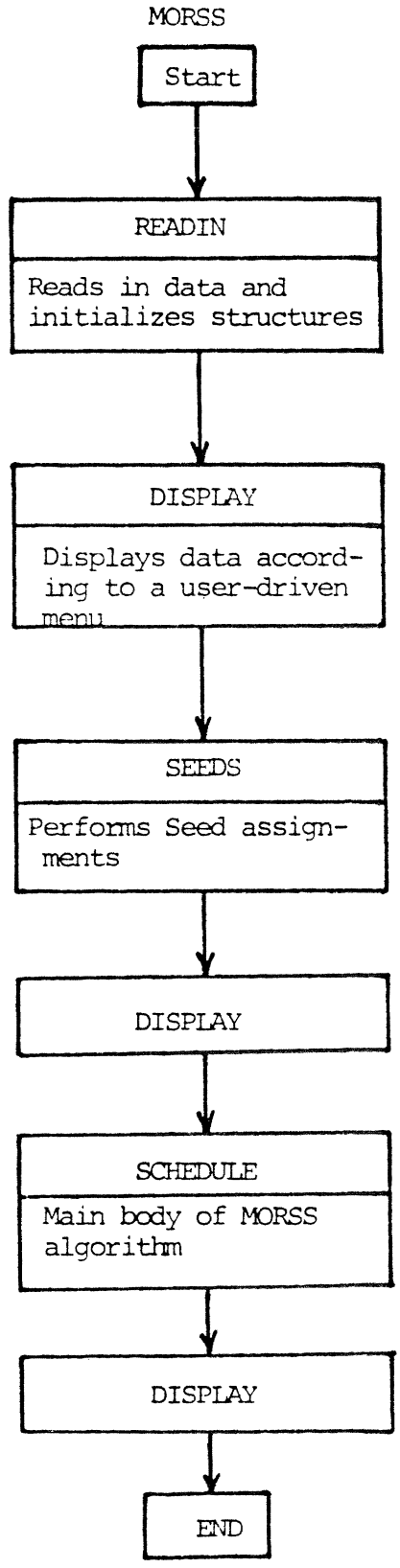

MORSS
Start
READIN
Reads in data and initializes structures
DISPLAY
Displays data according to a user-driven menu
SEEDS
Performs Seed assignments
DISPLAY
SCHEDULE
Main body of MORSS algorithm
DISPLAY
END

Figure 1.2

overall utility.

(b) MORSS incorporates queueing effects through port congestion utility. A more refined representation of queueing delays can be readily incorporated into the cargo lateness utility function (see also Chapter 3).

(c) MORSS is flexible. It can handle practically any number of cargoes per ship. Its hierarchical and modular structure allows it to adapt easily to the staggered availability times of different cargoes and ships. The code is easily expandable and maintainable, and its design is such that it can be easily "interrupted" to allow for interaction with the human scheduler.

(d) MORSS can handle many problem complexities easily. In fact, the addition of these may actually reduce subproblem size and overall running time. Such complexities include the presence of divisible and indivisible cargoes, route restrictions, cargo/port or other shipping constraints, changes in movements requirements, etc.

1.4 Computational Experience

We have tested the MORSS algorithm with data supplied to us by the MSC. The MSC database includes information on 505 cargoes, 232 ships and 26 ports.

Each of the cargoes is either a single item or a collection of items that have a distinct POE and a distinct POD. In addition, each cargo has a preferred ship type, an Earliest Pickup Time (EPT), an Earliest Delivery Time (EDT), and a Latest Delivery Time (LDT). Also known is that cargo's Short Tons, its Measurement Tons and its Deck Area. The 505 cargoes are classified into 8 categories, according to preferred ship type:

These are breakbulk (193 cargoes), seatrain (13 cargoes), RO/RO (151 cargoes), self-sustaining container (54 cargoes), tanker (45 cargoes), and barge carrier (25 cargoes) while 4 cargoes have an unspecified ship preference.

Ships are classified into 3 fleet types and 6 ship types: The fleet type codes are MSC controlled (38 ships), Ready reserve (38 ships) and Sealift Readiness Program (156 ships). The ship type codes are breakbulk (100 ships), seatrain (5 ships), RO/RO (5 ships), self-sustaining container (9 ships), tanker (25 ships) and barge carrier (18 ships). Information on ships includes capacity (weight/volume/deck area), draft, speed, cargo loading and unloading rates, as well as initial location.

Finally, the 26 ports of the MSC database are classified as follows: 12 are POE's, 5 are POD's, and the remaining 9 are both POE's and POD's. All ports of the database are located in the United States, the Panama Canal zone and the Pacific. Information on ports includes the throughput characteristics of their various berths and terminals. All inter-port distances are also known.

Given a particular "problem instance" (that is, a set of prescribed inputs for the MSC operational problem), we have assumed that the scheduler would like to obtain a preliminary idea regarding whether available resources are enough to satisfy the prescribed cargo movement requirements. If the opposite turns out to be the case, it would make little sense to proceed with a detailed scheduling run, because most cargoes would be delivered late. Instead, it would then make sense to relay infeasibility information immediately to the chain of command "upstream", so that either the cargo movement requirements are modified, or additional resources are

made available, or some other measure is taken to alleviate this problem.

Gross feasibility analysis can be performed at various levels of detail. At the simplest level, one can check whether the prescribed cargo delivery time requirements alone are reasonable or unreasonable. Such a test calculates the maximum time slack between each cargo's actual delivery time and its LDT under the optimistic assumption that the cargo leaves its POE immediately at its EPT and travels directly to its POD. The same test also calculates the minimum ship speed required if the cargo leaves its POE promptly, is delivered to its POD at its LDT and travels directly.

Applying such a test to all 193 cargoes belonging to the breakbulk category showed that delivering all of these cargoes on time is virtually impossible, irrespective of both actual ship resources and scheduling strategy. Similar observations were made in the MSC database on virtually all other cargo/ship categories. From our discussions with MSC personnel, we understood that such infeasibility of the database could be attributed in part to the "sanitization" process that was used to convert the database from classified to unclassified. Other factors might be valid as well, such as the facts that in practice a cargo's EPT and LDT are sometimes (if not always) determined by two different (and sometimes unrelated) decision processes. As a result, a cargo's LDT usually reflects neither that cargo's availability at its POE, nor its POE/POD distance.

A second-level feasibility test takes also into account ship resources as well. It formulates the feasibility problem as a "transportation" problem. It sets up a bipartite network, with supply nodes representing ships, and demand nodes representing cargoes. The optimal value of this problem is a lower bound on the actual total weighted

tardiness (ton-days late) that would incur under any actual allocation of available ship to cargoes.

We chose a subset of the MSC database for the initial runs of MORSS with certain goals in mind. For ease of analysis we wanted ships and cargoes to be of compatible types. We desired the problem size to be at once small enough to easily manage and comprehend, and large enough to contain the complexities of the MSC operational sealift problem. Finally, we wanted a nontrivial problem, one which reflected to some extent the infeasibility of some of the time windows in the MSC database.

Our objective in the initial comnputational study was twofold. First, we wanted to verify that MORSS was operating correctly and in a manner consistent with its design. Secondly, we desired to begin calibration of the parameters of the model, that is, empirically test the effect of seed assignments, cargo size and time window arrangement on the quality of the overall solution. Our procedure was also two-fold. First, we tested MORSS on increasingly larger problems until we were satisfied that the model was debugged. Then we began the calibration study, which we now discuss.

The initial runs confirmed our intuition that seed assignments greatly influence the quality of the final schedule. Our strategy was to try a variety of seed assignments in order to gain a measure of the quantitative effects of this key factor. These assignments were made randomly at first, then varied to test various hypotheses about what constitutes "good" seed assignments. In particular, we attempted to improve the quality of the solution (e.g. decrease the number of undelivered cargoes) by modifying the assignment of seeds from one run to

the next.

From the runs of MORSS, (see details in Chapter 4), it became evident that the performance of the system depends highly on the particular data set being used. Many factors are involved. Boundary conditions, cargo size, seed selection and interactions betrween large cargoes may account for much of the difficulty. In addition, the parameters involved in the utility functions need detailed calibration themselves. To aid in the analysis of the run, MORSS computes a large variety of statistics at the end of each run. These statistics are the measures by which we evaluated the performance of the system. The most important of the cargo statistics are the percentages of cargoes (delivered) on time and tons (delivered) on time. These most clearly reflect the primary concern of the MSC to deliver cargoes on time.

Among the ship-related statistics, route circuity, ship utilization and percent of time spent in port are the most important. By "route circuity" we mean the ratio of the total distance traveled by a ship divided by the sum of POE-POD distances for all cargoes carried by the ship. Thus, a low circuity (less than half) means the ship carries many cargoes most of the time, while a high circuity (more than 1.5) means the ship carries few cargoes at a time, and takes a rather circuitous route, probably deadheading a good part of the time also. These measures are important as indications of the efficiency at which the system operates.

Results of these runs show that ship utilization and the percentage of tons delivered are not correlated with the percentage of tons delivered on time. They also show a strong linear correlation between the percentage of on-time cargoes and tons. The percentage of cargoes on

time seems close to being independent of the percentage delivered. These preliminary results indicate that ship utilization is not a good performance measure for problems with tight or infeasible scheduling requirements. This result will be checked more closely with a larger feasible set of cargo movement requirements during the calibration procedure.

In general, our initial MORSS runs appear to be interesting and encouraging. The algorithm generates schedule patterns that seem to be "clever" with respect to a particular cargo movement requirement. For instance, roundtrips or other, more complicated routes are generated, and large-scale cargoes are split. Further interpretation of these and other test runs would feed back into further refining the algorithm.

1.5 Miscellaneous Other Activities

Other than the development of MORSS, there have been a number of other activities related to the project:

(1) We have had four meetings with MSC personnel: One in Washington, D.C. in May of 1983 where a general discussion (which included the University of Pennsylvania team) was conducted, and three at MIT, in December of 1983, in November of 1984 and in May of 1985, where representatives of the MSC interacted with the MIT team on topics related to the algorithm and the MSC database. The last two meetings included computer demonstrations by the MIT team of versions of the MORSS algorithm.

(2) In March of 1984 the MIT team presented its work at a workshop organized by the MSC and the Joint Deployment Agency, in Tampa, Florida. The workshop included presentations by the University of Pennsylvania and Georgia Tech groups (see also Jarvis, et al, 1982) on their pojects.

(3) Progress in the project has been also presented at three ORSA/TIMS meetings: One in Orlando (Orlin and Psaraftis, 1983b), one in San Francisco (Orlin and Psaraftis, 1984), and one in Boston (Psaraftis et al, 1985).

(4) A student M.Sc. thesis was written on the solution of a version of the MSC problem using an insertion algorithm (Jeng, 1984).

(5) A PhD thesis was submitted on the development and analysis of certain algorithms, exact and heuristic, for "analyzable" version of the MSC problem (Kim, 1985). A paper based on this work was presented at the EURO-VIII Conference in Bologna, Italy (Kim et al, 1985).

(6) An annotated bibliography of about 70 references in this general problem area was prepared (Thompson, 1983). These references are maintained in the MIT project library.

(7) Finally, contacts have been maintained with a sister project, sponsored by the Military Airlift Command (MAC), at MIT's Flight Transportation Laboratory. Although the scheduling problems of MSC and MAC and the scopes of the two projects are different, both sides are interested in exploring the potential for transferring scheduling methodologies from one project to the other.

1.6 Current and Future Plans

Further research directions entail six tasks: (1) investigate and calibrate alternative utility functions, (2) investigate cargo assignment interactions, (3) develop more sophisticated seed selection methods, (4) model queueing effects at ports, (5) undertake sensitivity analysis and (6) further test the algorithm and enhance its user-friendly features. At

the time of the writing of this report, the MIT team is concentrating on tasks (1) and (6), all other tasks being left for a future phase of the project. A detailed description of these tasks is given in Chapter 5.

CHAPTER 2

BACKGROUND ON THE PROBLEM

2.1 Introduction

Before one can attempt to solve a problem, one must first define it. To that end, the MIT project team has spent a considerable amount of effort in the early phases of the project in order to better understand the structure and complexity of the problem at hand. A second, parallel effort concerned the definition of "the problem" in clear and explicit rather than vague or ambiguous terms.

As in many complex problems, ambiguity in the MSC problem ranges from issues related to lack of a clear definition of the problem objective(s) or constraints, to issues such as whether a specific problem parameter must be considered a decision variable or an exogenous user input. To state a few examples, issues that have to be addressed to resolve ambiguity include (but are not limited to) the following: Does the MSC wish to maximize the number of cargoes delivered on time, or minimize the total ton-days of cargo delivered late? Does one allow a cargo to be split? Does one consider due dates as "soft" or as "hard" constraints? Does one, in fact, consider due dates as decision variables in this problem, or does one take them as exogenous user inputs? (With respect to this last issue, it is clear that somebody in the whole chain of command in sealift scheduling somehow makes a decision regarding what the due date of a cargo should be; so the question is whether this decision is part of our problem). Does one similarly consider ports of embarkation or debarkation as decision variables, or take them again as exogenous user inputs? It is

clear that the resolution of such ambiguities is paramount in our ability to create a realistic model for the real-world problem and ultimately solve the problem.

Both efforts described above (that is, complexity assessment and problem definition) were undertaken in parallel because we wished to come up with a problem definition that was both realistic and at the same time well-understood in terms of structure and complexity. We paid particular attention to the identification of similarities between the problem at hand and other problems that were tackled in the past, so that we could take maximum advantage of known techniques that might be transferred to this problem.

The purpose of this chapter is to describe the results of our assessment on that score, and, in a way, put the problem at hand into the proper perspective. Such a perspective forms the basis of the development of the MORSS solution procedure that will be described in Chapter 3. An overview of the real-world problem and current practices is presented in Section 2.2. Section 2.3 reviews similarities and differences between this problem and other related problems and explores the possibilities of methodology transfer among these problems. Section 2.3 specifies the version of the problem to be examined and distinguishes between parameters that are explicit decision variables in this analysis, and other variables, which are exogenous user inputs for the version of the problem at hand. Finally, Section 2.5 sets forth a set of generic design features that a computer-assisted procedure should possess in order to be able to solve the MSC operational problem, and comments on the suitability of the SAI algorithm for solving that problem.

2.1 Overview of the Real-World Problem and Current Practices

The Military Sealift Command (MSC) is the agency responsible for providing sealift capability for the Department of Defense. Three of its most important missions according to Scott (1982) are the following:

(1) Provide peacetime logistical sealift support of military forces worldwide;

(2) Develop plans for the expansion of the peacetime sealift fleet to support military contingency operations and mobilization. And

(3) Acquire and operate this expanded fleet to provide contingency and mobilization sealift support of military forces worldwide.

The MSC scheduling activities in support of peacetime logistical support are similar to those accomplished by commercial liner companies and by chartered shipping operators. The problem is to size the fleet at the optimum level and to schedule ships to transport cargoes among ports most economically.

The scheduling requirements for the MSC contingency and mobilization mission areas are different from those during peacetime operations. Besides involving a larger number of ships and cargoes, (this number can be as high as several thousand cargoes and 1,500 ships) the ability to deliver cargo on time now becomes paramount. Under such a setting, the objective of the MSC is to ensure that all cargo, dry and liquid, arrive at destination as planned.

The key objective of MSC's strategic planning of sealift operations for contingency and mobilization situations is to move military forces and supplies to the required location during a period of potential or actual

conflict within a required time. For this reason, MSC has initiated the development of a comprehensive methodology, SEASTRAT, to perform the scheduling of MSC transportation resources and thereby evaluate the feasibility of meeting mobilization requirements. At present, SEASTRAT is intended to assist planners solve the "deliberate planning" version of the MSC routing and scheduling problem. This deliberate planning problem significantly differs from the MSC "operational" scheduling problem, which is the problem on which the MIT project has focused.

Before we proceed with SEASTRAT, we highlight the similarities and differences between these two classes of problems:

Both the deliberate planning problem and the operational scheduling problem call for an assignment of cargoes to ships so as to satisfy "as best as possible" the cargo movement requirements and the due dates that have been specified as part of an "Operational Plan". In the deliberate planning problem, inputs are generated according to a plausible scenario that represents a contingency that may arise in a part of the world. In the operational problem, inputs correspond to a real scenario that has actually occurred and evolves in time.

Despite their conceptual similarities (e.g. both problems essentially call for a "reasonable," "good," or "optimal" allocation of cargoes on ships to satisfy the due dates that have been specified), the two problems have significant differences: For instance, the emphasis in the deliberate planning problem places more emphasis on the determination of the minimum number of ships that are necessary for the successful execution of a plan, while the operational problem emphasizes the efficient use of available ships to meet the due dates with the minimum amount of

delay. The deliberate planning problem involves relatively long time horizons (and thus involves a greater degree of uncertainty; initial ship positions, queueing delays at ports, and other factors are either not known with certainty or cannot be predicted with accuracy) while the operational problem is typically of shorter duration (and thus involves a lower degree of uncertainty; most of the inputs are either known a priori - e.g. positions or availabilities of ships, or become eventually known to the decision-maker along the course of events). Information in a deliberate planning problem is typically highly aggregate, while in an operational problem one is typically faced with highly detailed information on the problem. Several other differences exist, the enumeration of which is outside the scope of this report. In our opinion, the most significant conceptual difference between the two problems is as follows. Whereas the data to the deliberate planning problem is specified in advance and is static, the input data of the operational problem changes dynamically in time, with the word "dynamically" interpreted as "at the same time the decision-making process (whether automated, manual, or man-machine) is taking place". For instance, new cargo requirements may be imposed a week after the occurrence of the scenario, that is, <u>after</u> the initial cargo-to-ship allocation decisions have been made. Or, certain ships and/or ports may cease to become operational for various reasons (malfunction, attrition, etc). Many other examples can be devised. It is therefore clear that special consideration should be given to the scheduling methodology that "solves" the operational problem, so that its dynamic nature is taken into account. Similarly, extreme caution should be exercised in attempting to "transfer" to the operational problem

methodologies that have been developed to solve the deliberate planning problem (more on this point later).

With these considerations in mind, we come back to SEASTRAT and describe how it relates to current MSC practices. The objective of SEASTRAT is to schedule ships and cargo transport so as to deliver cargo to the required ports of debarkation within the proper time (see SEASTRAT, 1981). The ship routes and schedules must be consistent with initial cargo and ship locations, port capacities, cargo and ship types so as to provide a feasible (and hopefully "optimal") solution to the scheduling problem.

SEASTRAT will eventually replace the Strategic Sealift Contingency Planning (SEACOP) system. SEACOP is, by today's standards, a rather old system, run for the first time on MSC's Honeywell 1200 computer in 1972. It was designed to perform detailed planning. It takes into account precise ship characteristic data (e.g. exact speed, capacity, etc.) and produces a detailed shipping schedule. SEACOP was not readily accepted by the planning community, which was at that time oriented to gross feasibility planning. The planning philosophy has since changed, and the detailed output of SEACOP corresponds to current requirements. However, there are a number of deficiencies in the system that cause problems. The operation of the system consumes considerable computer time (up to forty hours for large plans); there is no restart capability; the loading and scheduling algorithms lack credibility. There is also a recognized need for an automated transportation planning, which is not provided by SEACOP.

According to Kaskin (1981) most of the problems that analysts have discovered about SEACOP are due to deficiencies in the logic within the Schedular Subsystem. There are also several other problems that are the

result of defects in the overall system design and implementation.

When a Commander-in-Chief develops an Operational Plan, he also establishes a Time-Phased Force Deployment Data (TPFDD) file (see SEASTRAT, 1981). The file contains information about what cargo is to be transported from what port of embarkation to what port of debarkation, with earliest and latest arrival dates specified.

Currently the MSC processes the TPFDD by means of the SEACOP system. The TPFDD is edited for validity of data, and movement requirements are aggregated on the basis of common cargo type, ports, and time requirements. The system then calculates ship availability by taking into account the level of mobilization, determining what ships are usable under that level (e.g. owned fleet, available at all times; NATO fleet, available only in NATO plans; etc), and calculating the time required to travel to various ports and discharge their cargo. Feasibility is then assessed by simulating the loading of ships and the transport of goods, and then determining which loads were not scheduled or arrived later.

Under the present system, the automated feasibility analysis must be improved by hand manipulation (see SEASTRAT, 1981). For example, one of SEACOP's deficiencies is that it can only load ten aggregated cargoes onto a ship, no matter how small the resulting load is. It is therefore obvious that a process of hand-scheduling cargo into the empty space or underloaded ships may show a plan to be more nearly feasible than the model has indicated. Some analysts try to improve the detailed schedules produced by SEACOP by hand-rescheduling or by proposing changes in times associated with specific loads. However, this approach becomes impractical with large plans. Other analysts make a broader set of suggestions about relocation

or rescheduling of cargoes, after looking at summary information output by the model (number of shiploads per port, etc.). An analyst can prepare specific changes to the TPFDD and run SEACOP again to check feasibility of the modified transportation requirements. However, due to the SEACOP's inefficiency (up to forty hours required for large plans), any process of successive approximation to a feasible plan is severely limited.

A more accurate and more efficient method for conducting feasibility analysis is needed to be employed by SEASTRAT. This method must be able to identify which requirements result in infeasibility. In this way, changes that would render a plan feasible could be suggested. Such changes could include changing the level of mobilization, increasing shipping capacity in specified ways, suggesting a better distribution of cargoes among ports of embarkation, etc. If this method for suggesting changes involves successive approximations (a repeated sequence of adjusting the TPFDD and performing feasibility analysis), then the feasibility analysis must be a quick process that can be repeated several times without affecting the overall analysis schedule.

Unfortunately, the schedules produced by SEACOP are not realistic. According to Kaskin (1981) "too many ships with less than 50% utilization deliver their cargoes late." This poor performance has to do with the design of SEACOP's Schedular Subsystem, as we shall see below. The Schedular assigns a cargo movement requirement from the TPFDD file and scores each ship eligible to carry the requirement to the port of debarkation, by using a simple formula. (An eligible ship is one that meets the port of debarkation draft and length restrictions, that has sufficient boom capacity, and that will allow the ship to meet its latest

arrival date). The score of each eligible ship depends on the ship's expected arrival date to the port of debarkation and on the ship's utilization factor (i.e., capacity of ship utilized as a fraction of ship's total capacity). The eligible ship with the lowest score is assigned the cargo.

SEACOP's scoring formula does not schedule ships against movement requirements in a realistic manner. It determines the fate of a ship upon assignment of its first cargo without considering other available cargoes. Additionally, if a cargo is assigned to a ship that does not, after all cargo movement requirements are considered, have sufficient cargo to be commited to sail, then such cargo is not sent. In reality, one could possibly reassign the cargo to another ship that might arrive a little late. Another flaw, mentioned already, is that the Schedular currently does not allow more than ten movement requirements to be assigned to the same ship. Since movement requirements are often small in size, ten loads may not be enough to fill the ship above the minimum required for that ship to sail. Thus, the cargo will become "frustrated".

Kaskin in his Point Paper concerning suggestions for SEACOP's improvement (Kaskin, 1981) states the following:

> "The overall result of the above deficiencies is that SEACOP produces unacceptable schedules for most plans. This can be verified by talking with the analysts who use the model daily. Unfortunately, there is presently no way to determine how poor the SEACOP schedules really are. SEACOP has never been validated. That is, SEACOP's outputs have never been compared with the best schedule that human operators might come up with, given the same ships and movement

requirements".

To date, the most significant algorithmic development effort in conjunction with SEASTRAT (that is, with the deliberate planning version of the problem) has been the work of Science Applications, Inc. (SAI, 1982). SAI developed an algorithm called "Scheduling Algorithm for Improving Lift" (SAIL).

SAI formulated a transportation network model to solve the deliberate planning problem. The model was designed to minimize total system costs (the cost of the ship use and penalties for lateness of cargo delivery represented as a function of the cargo/ship assignments) subject to constraints which required that all cargo be delivered and ship capacity not be exceeded. The objective function coefficients incorporated all system costs, including those of delays. Then the problem was solved by successive iterations over the values of those coefficients, using a well-known solution algorithm for the transportation problem, until no further improvement could be made. Throughout the procedure, time constraints were regarded as being "soft". (If a time window can be violated, it is referred to as a "soft" time constraint). The method of updating cost coefficients and assigning seed cargoes was also heuristic.

A detailed description of the SAI methodology is beyond the scope of this report and can be found in SAI (1982). However, a crucial question that the MIT team felt obliged to answer was to what extent the SAI methodology, which was developed for the deliberate planning problem, could be "transferred" to the operational problem. This report addresses this very important issue in Section 2.5.

We conclude this section by noting that to date, there has been no

counterpart of SEACOP or the SAI algorithm for the operational MSC problem. Thus, if an actual mobilization situation were to occur, detailed sealift scheduling would have to be done by hand.

2.3 Relationship to Other Problems

To gain further insight into the structure of the MSC problem, we now review some problems in other environments that are conceptually related to this problem, and thus, conceivably are promising from a methodological viewpoint.

(1) The Dial-A-Ride Problem

The dial-a-ride problem is the problem of carrying customers from distinct origins to distinct destinations within specified pickup and delivery time windows. The problem can be considered as a restricted version of the MSC problem, in which all ships and cargoes are assumed identical and no ship/cargo/port restrictions exist. Whereas it is clear that such a restriction is unrealistic, we outline below the main algorithmic developments with respect to dial-a-ride over the past decade.

Several approaches have been developed for solving versions of the dial-a-ride problem. Wilson et al, (1976, 1977) and Bodin and Sexton (1982) developed heuristic algorithms for practical applications. Psaraftis (1980, 1983a, 1983b, 1983c) analyzed and developed exact and heuristic algorithms for several different versions of the problem. Recently, Jaw et al (1982, 1984) developed heuristic algorithms for multi-vehicle problems with time constraints.

The work of Psaraftis (1983a) was an exact dynamic programming algorithm with forward recursion, whose time bound was $O(n^2 3^n)$ for a single

vehicle problem of n customers. The objective of the problem was to minimize the time needed to service all customers and the algorithm identified infeasible (with respect to time constraints) problem instances.

Bodin and Sexton (1982) formulated a mixed-integer linear programming problem for a one-sided time constraint (desired delivery time) problem and developed an algorithm based on Benders' decomposition for solving it. The objective was to minimize the total "inconvenience" customers may experience due to excess ride time and deviation from the desired delivery time. The initial solution for the algorithm was obtained by the "space-time heuristic", which was a variant of the "nearest neighbor" heuristic in which the "measure of closeness" was represented by a parameter called "space-time separation". Later, Bodin and Sexton, (1982b) developed a procedure to solve the multi-vehicle version of the problem. In this procedure, a "swapper" algorithm was used to improve vehicle clusters and the "space-time" heuristic was used as a single vehicle routing subroutine.

Jaw et al (1982) developed an algorithm for solving the multi-vehicle dial-a-ride problem with "soft" time-window constraints. The objective of the problem was to develop a set of routes for a fleet of vehicles serving customers who have to be picked up from specified origins and delivered to specified destinations so that overall vehicle productivity was maximized. The algorithm consisted of three successive and distinct steps: "grouping", "clustering", and "routing". Grouping divided customers into "time groups" on the basis of their desired pickup and delivery times. Clustering separated customers of each time group into "clusters" and assigns vehicles to serve each cluster. "Routing" generated

routes for each individual vehicle to serve every cluster in turn and for every time group.

In Jaw et al (1984), the same authors developed an "insertion" algorithm for the "hard-time-constraint" version of the problem. In this version, the solution which violates any of given time constraints is regarded as "infeasible". The algorithm sequences the customers in ascending order of their earliest pickup times. Then, customers are assigned to different vehicles based on certain criteria. The current requests are inserted into the previously built-up tour of each vehicle in service. Among the feasible insertions, the one which optimizes the objective value is chosen as the best insertion. An insertion is infeasible if, as a result, any service constraints are violated for the current request or for any of the customers already on board the vehicle.

This procedure has some potential for solving the sealift problem. In fact, Jeng (1984) investigated the potential of "transferring" this methodology by developing the framework for a sequential insertion algorithm for the MSC problem.

It is important to point out that, with the exception of the dynamic programming approach developed for the single-vehicle problem and which is viable only for very small problems, all other dial-a-ride algorithms have been heuristic, especially the ones developed for the multi-vehicle problem. Given the fact that the MSC problem is a generalization of (and hence, more difficult than) the dial-a-ride problem, it is unlikely that the general version of the MSC scheduling problem can be eventually solved efficiently by an exact (optimization) approach. Moreover, any heuristic (i.e. approximate) methodology that is developed

for this same general version of the MSC scheduling problem is very unlikely to lend itself to worst-case or average-case analysis. Such analyses measure how good (in terms of deviation from the theoretical optimum) a particular heuristic algorithm is on a worst-case basis or on the average (respectively), and can be typically developed for simpler, more restrictive versions of a given problem. For the MSC case, such analyses are very interesting from a theoretical point of view, but less interesting from a practical point of view because the versions that are amenable to such an analysis are very restrictive. In any event, such versions have been systematically analyzed in the Doctoral dissertation of Kim (1985) and are briefly reviewed in Chapter 5 of this report.

(2) The Bulk Delivery Scheduling Problem

Another problem that resembles a special case of the MSC problem is one analyzed extensively by Fisher et al (1982). They considered the problem of scheduling a fleet of vehicles delivering a bulk product stored at a central depot. The objective of the problem was to maximize the value of the product delivered to all customers, less the fleet operating costs incurred in making these deliveries. They developed a mixed-integer programming formulation of the problem and a solution algorithm based on Lagrangian relaxation and a multiplier adjustment method.

The algorithm first heuristically generates a menu of possible vehicle routes taking into account the geographical location of customers, and the amounts of demands and truckloads. A route is excluded if the customers on the route are spread out through a large geographical area or if the amount of the product that could be delivered to the customers on the route is significantly less or significantly more than a truckload.

A model is formulated to select optimally from this menu of possible routes a subset that could actually be driven, specifying the time each route should start, the vehicle to be used, and the amount to be delivered to each customer on the route. In that case, the request can either be turned down or another back-up vehicle can be added into the system. Also, the algorithm has been modified to accept vehicle capacity constraints. Then, as a technique for solving the problem, a Lagrangian relaxation method is used introducing a multiplier adjustment method.

The Lagrangian relaxation method is an important computational technique for solving certain mixed-integer programming problems. The rationale underlying the method is the fact that many hard combinatorial problems can be viewed as easy problems complicated by a relatively small set of side constraints. Dualizing these side constraints (weighing them by multipliers and placing them in the objective function) produces a Lagrangian problem that is easy to solve and whose optimal value is an upper bound (for maximization problems) on the optimal value of the original problem. Thus, it can be used in place of linear programming relaxation to provide bounds in a branch and bound algorithm.

Recently, these same authors have adapted this method to the "peacetime tanker scheduling" problem of the MSC (Fisher and Rosenwein 1984). This problem can be considered as counterpart of the MSC emergency scheduling problem in the sense that the objective in peacetime operations is similar to that of a commercial shipping company (i.e. minimization of costs) rather than the timely delivery of the cargo.

One key conceptual difference between the truck problem and the tanker problem is the fact that whereas each truck's trip starts and ends

at the same prescribed point (the depot), the endpoints of each of the tankers in the fleet are neither the same, nor necessarily prescribed in advance. Tankers do not have to return to a central depot at the end of their trip. Indeed, their trip may consist of an open-ended sequence of port visits. Fisher's team handled this variation from their original model by imposing a limit on the amount of idle time each tanker could wait at each port and by decomposing the problem and looking at one "time horizon" at a time. For instance, they could schedule ships within the next (say) 60 days (that is, ignoring scheduling demands from day 61 on) with the additional requirement that no ship should idle at any port by more than (say) a week. Such a treatment served to limit the number of schedules generated by the schedule generator to a manageable number.

Fisher's team reported satisfactory computational experience with this procedure for a series of test problems (Fisher et al, 1984). This suggests that this approach is certainly promising for the peacetime tanker scheduling problem of the MSC. However, the extent to which this methodology can be extended to solve the MSC operational problem is less clear. The MSC operational problem involves in addition to oil (which is a bulk commodity) general cargo as well, either containerized or breakbulk. This means that a ship is likely to carry a number of different cargoes at the same time. This factor alone may make the number of schedules needed to be generated by the schedule generator prohibitively large. Moreover, it is not clear how this procedure can handle dynamic updates of input data. Thus, in addition to a significant amount of new reserch that would have to be undertaken to address these and other issues, we conjecture that this approach would have to undergo a significant degree of redesign, most

of it heuristic, to be tailored to the features of the MSC operational problem.

2.4 Definition of the Operational Problem; Assumptions

As stated in the project proposal (Psaraftis and Orlin, 1982), an oversimplified and generic definition of the MSC operational routing and scheduling problem is the following:

> Given a set of available ships (located at any given point in time at given points somewhere in the ocean) and a set of cargoes (awaiting pickup at given ports of embarkation (POE's) and requiring delivery at given ports of debarkation (POD's) and within specified time limits) that is the allocation of cargoes to available ships that "optimizes" a prescribed delay-related measure of performance?

After extensive discussions both within the MIT team and with MSC personnel, the following additional clarifying assumptions have been made regarding the definition of the problem at hand:

(1) The problem is dynamic in nature, that is, information on inputs such as cargoes, ships, and ports may (but does not necessarily have to) become available to the decision-maker (the scheduler) concurrently with the decision-making process. Previous information may be updated.

(2) The problem is deterministic in the sense that no probabilistic information on the input variables of the problem is available. Issues such as queueing and congestion at ports which are inherently probabilistic are definitely taken into consideration, but in an approximate, and deterministic fashion at this stage of the research. As will be described in Chapter 3 the queueing process is a very important area for further

research, (see also Chapter 5).

(3) We assume that all decisions concerning the mode of transport of a particular cargo have been already made at the strategic level. That is, this problem is not concerned with deciding on whether a particular cargo should be carried by the MSC or by MAC (the Military Airlift Command). Instead, the problem looks only at cargoes that have been already assigned to the MSC for transport. Similarly, this problem is not concerned with interactions between the MSC and other Transportation Operating Agencies of the Department of Defense, such as MTMC (the Military Traffic Management Command), or others. We assume that for each cargo, its POE, its POD, and the time limits within which its transport should take place are user inputs rather than decision variables. In other words, deciding which is the most appropriate POE for a particular cargo, or determining when this cargo is available at its POE or due at its POD are not part of the problem at hand. These important decision issues are currently being investigated by a Georgia Tech project, sponsored by the Joint Deployment Agency (see Jarvis et al, 1982). These issues are outside the scope of our project.

(4) We allow for multiple types of ships and cargoes (i.e. tankers, ro/ro ships, breakbulk, etc.), as well as for the splitting of a cargo (usually due to size), but our methodology does not consider transfers of cargoes among ships. The implication of this is that feeder operations which would bring a certain cargo by ship into a certain port, to be picked up subsequently by another ship, are not modeled by our problem. We only consider such cargoes (if any) after they have arrived (by whatever mode) at their designated POE's.

(5) We assume that each ship can carry any number of distinct cargoes

(as long as they are compatible and its capacity is not exceeded) and that it can make any number of stops on its schedule (to pick up and/or deliver other cargoes). Pickups and deliveries can be interspersed along the route, and multiple roundtrips or triangular routes can be considered.

(6) This problem does not consider ship convoys, nor cargo precedence constraints of the form "this cargo should be picked up before this other cargo is delivered" or "cargo # 37 should arrive before both cargoes # 1 and # 48 arrive." Issues such as the above might be very important in practice, but were left for a future phase of this work.

(7) Each cargo is assumed to have a POE, a POD, an Earliest Pickup Time (EPT), an Earliest Delivery Time (EDT) and a Latest Delivery Time (LDT). Also the data include the direct distance between that cargo's POE and POD, its weight, volume and surface area. After extensive discussions with MSC personnel, we decided that EPT's and EDT's should be considered "hard" constraints, whereas LDT's should be considered "soft". This means that a cargo cannot be picked up before it becomes available at its POE (at its EPT) and also cannot be delivered at its POD before its EDT. This last requirement was imposed because there might be valid logistical reasons on why a certain cargo cannot be delivered earlier than a prescribed time (lack of adequate support facilities etc.). With regard to a cargo's LDT, we decided that it should be considered a "soft" constraint (hence, is amenable to violation) because we judged that it would be better to deliver a cargo late (especially if it is only a few days late) than not deliver it at all (which could happen if this constraint also were "hard"). At the same time, and so as to discourage late deliveries, we decided to incorporate a term into our objective function that penalizes cargo

tardiness (see also Chapter 3). Our model also allows for the possibility of a cargo not being delivered. In such an event, that cargo would be "flagged", and its due date might subsequently be readjusted after an interaction between the MSC and the authorities responsible for issuing the cargo movement requirements.

(8) The data for each ship includes quantities such as capacity (weight/volume/deck area), draft, speed, cargo handling capacity and initial location. The data for each port includes the draft and the throughput characteristics of its various berths and terminals.

(9) Finally, we briefly discuss the objective function of this problem (more details will be presented in Chapter 3). From our discussion with MSC personnel we concluded that there are three primary events that are likely to create problems in any given operational scheduling situation, and hence, should be explictly considered by our approach: (a) Late delivery of cargoes, (b) low ship utilization and (c) severe port congestion. The occurrence of any one of the above three events is considered an undesirable outcome in any operational situation, and therefore should be explicitly penalized. In SEACOP, (a) and (b) have been recognized to cause problems (Kaskin, 1981), whereas (c) has been completely neglected. Of course, in reality (c) is ultimately reflected into (a). However, we thought important to consider queueing and congestion at ports explicitly rather than implicitly because of the broader ramifications that this issue could create in the logistics of the overall problem. We present the modeling of these three criteria in Chapter 3.

2.5 Generic Design Features of an Operational Scheduling Algorithm (MIT Approach)

Based on all the previous considerations, and after a significant degree of interaction with MSC personnel, the MIT team came up with a set of generic design features that a computer-assisted procedure should possess in order to be able to solve the MSC operational routing and scheduling problem. These features are the following:

(1) It is essential that such an algorithm be interactive. One should always have the "human in the loop" and enable him/her to override the computer at will. Various options should be designed, ranging from a completely "manual" approach where all major allocation decisions are made by the human operator, to more sophisticated modes where the computer deals with more difficult problems (e.g. routing) but still allows user discretion for "key" decisions, or perhaps even to a fully automated mode where the computer makes a number of "default" assumptions and solves the whole problem with no user intervention.

(2) The algorithm should have a "restart" capability, that is, should be able to efficiently update schedules at any time within the execution of a plan, without compromising decisions already made. In particular, new cargoes should be able to be "inserted" quickly into existing schedules, cargoes or ships that are causing problems should be able to be "deleted", etc. Efficient list-processing techniques (available at such programming languages as PASCAL or PL1) should be implemented for fast database manipulations.

(3) The algorithm should be hierarchically designed, that is, allow the user to start the decision-making process with "first-cut" gross

feasibility analyses (possibly in several levels of aggregation) and then proceed with aggregate scheduling (that takes into account only the most significant and best-established problem factors, but ignores or simplifies other factors that are difficult to nail down - such as queueing delays). Ultimately, the program would solve the whole problem where all factors, from the most aggregate to the most detailed, are incorporated. Such a feature is considered important because many significant insights may be obtainable without having to solve the whole problem (e.g. a "quick and dirty" feasibility analysis may establish that some due dates are infeasible and hence allow the user to inquire for adjustments before further decisions are made).

(4) Finally, we consider it important that this algorithm be user-friendly. In particular, graphics aids are significant features that can enhance the efficiency of the man-machine interaction.

With the above considerations in mind, let us now reconsider the SAI methodology outlined in Section 2.2. It is of course fair to say that the SAI algorithm was not designed for the operational problem and therefore it would be unreasonable to expect that approach to work well in a setting different from the one it was developed for.

A detailed description of the SAI methodology is beyond the scope of this report. As stated earlier, SAI essentially formulates the deliberate planning problem as a "transportation" problem whose objective function incorporates all system costs, including those of delays, and "solves" that problem by successive approximations in the objective function. Those approximations are necessary since the actual objective function is (very) nonlinear while the "transportation" algorithm

can only handle linear forms. However, it is unlikely that final convergence of the assumed linear form to the actual objective can be always achieved or verified, due to the intrinsic nature of the objective function.

The rest of this section discusses whether the SAI algorithm (a) currently has the features outlined earlier and (b) if not, what changes would be necessary to incorporate such features:

(1) Interactive feature: SAI either does not have it, or has it in a very crude form. We expect that making the approach interactive would not be difficult conceptually, in principle, but that it would involve a substantial amount of new software development.

(2) Restart capability: SAI does not have this capability currently. Introducing the feature could be done to some extent, although it would involve some thinking as to how to make the algorithm "remember" previous assignments. More difficulty is expected whenever one or a few cargoes need to be "inserted" into existing schedules (actually, the SAI FORTRAN algorithm may be quite cumbersome in doing this). In all cases, substantial new software development would be necessary.

(3) Hierarchical design: SAI has that feature, but only to a certain extent. For instance, its first iteration is in itself a gross flexibility analysis. However, SAI falls short of incorporating some very important features into the problem. The most important of those is queueing at ports. It is not clear to us whether SAI can be modified so that queueing is explicitly taken into account.

(4) User-friendly/graphics feature: SAI does not have the feature, but it would be relatively straightforward to implement it.

It is clear from the above assessment that it would be rather difficult to modify the SAI algorithm to function in an operational setting. Similar conclusions can be reached regarding the other methodologies described in Section 2.4. of this chapter. Given that the prospects of an exact solution method are remote, the MIT team developed a heuristic procedure which is, by design, specifically tailored to the nature of this problem. This procedure is described in the following chapter.

CHAPTER 3

THE MORSS ALGORITHM

3.1 Introduction

MORSS is an acronym for MIT Ocean Routing and Scheduling System. It consists of four subsystems, READIN, DISPLAY, SEEDS, and SCHEDULE.

The first subsystem, READIN, creates and initializes all data structures and reads in all data. The second, DISPLAY, is a menu-driven, data management subsystem which enables and enhances the interactive nature of MORSS. The third subsystem, SEEDS, initializes the scheduling process by assigning a maximum of one cargo to each ship. The fourth subsystem, SCHEDULE, is the main functional part of MORSS. Its purpose is to form a schedule for each ship, based on the given cargo movement requirements. The flow logic for these subsystems is given in Figure 3.1. Details on the data structure design are given in Appendix A of this report. Details on the organization of the routines as well as flowcharts are given in Appendices B and C.

In the rest of this chapter we concentrate on the approach we have used the routines associated with SCHEDULE. We discuss the assumptions, problems, movitations and rationale which led us to structure these routines as we have.

3.2 The Scheduling Subsystem

This section contains a detailed description and explanation of the scheduling subsystem SCHEDULE. SCHEDULE operates after seed assignments have been made - (more on this in Section 3.5). Its function is to assign cargoes to ships so as to maximize the net overall utility of all

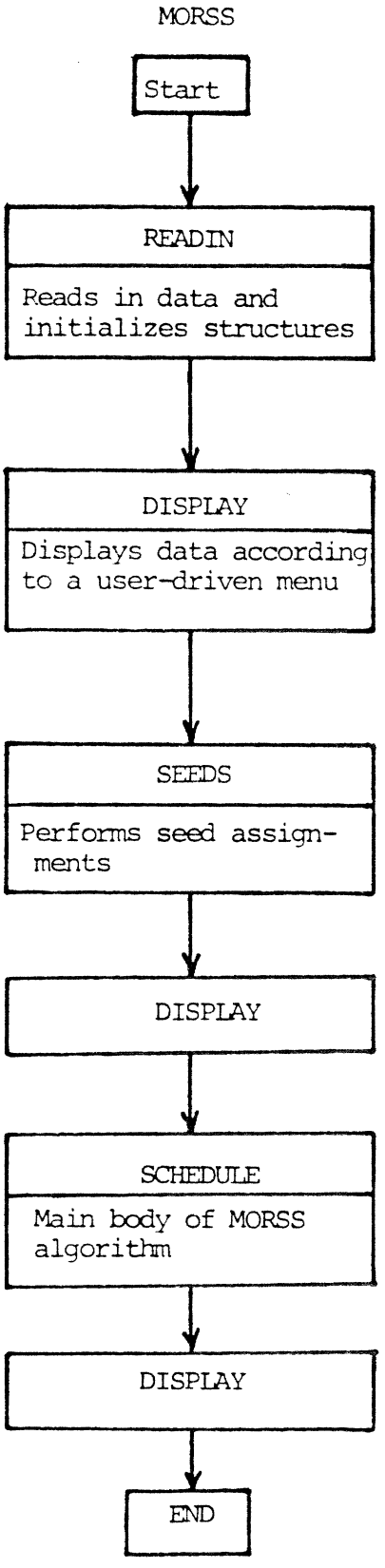

MORSS
Start
READIN
Reads in data and initializes structures
DISPLAY
Displays data according to a user-driven menu
SEEDS
Performs seed assign-ments
DISPLAY
SCHEDULE
Main body of MORSS algorithm
DISPLAY
END

Figure 3.1

assignments. It does this by following the sequence of steps outlined below (details follow):

Step 0: Initialize "master list" of unassigned cargoes.

Set up overall horizon (0,T) (T:user input)

Select length of individual time horizons L (user input: $0 \leq L \leq T$)

Select fraction a (user input: $0 \leq a \leq 1$)

Set $k=1$, $t_1=0$.

Step 1: Set up next time horizon (t_k, t_k+L)

Form list of cargoes eligible for assignment (all cargoes in "master list" whose EPT's are between t_k and t_k+L).

Step 2: Calculate assignment utilities for all eligible cargo/ship pairs (see section 3.2).

Step 3: Form and optimize a transportation network using assignment utilities as arc costs. Resulting assignment forms the "tentative assignment" for time horizon (t_k, t_k+L).

Step 4: Return (a) all unassigned cargoes by Step 3, (b) all tentatively assigned cargoes whose EPT's are between t_k+aL and t_k+L, and (c) all tentatively assigned cargoes which interact unfavorably with other assigned cargoes, into "master list" of unassigned cargoes. Make all other cargo/ship assignments in (t_k, t_k+aL) "permanent" and remove corresponding cargoes from "master list" of unassigned cargoes. Remove any "infeasible cargoes" (see Section 3.5) from "master list" of unassigned cargoes.

Step 5: "Roll" time horizon: Set t_{k+1} = Lowest EPT of all cargoes in "master list" of unassigned cargoes. Set $k=k+1$ and go to Step 1.

MORSS is based on a "rolling horizon" scheme (see Figure 3.2). The

ROLLING HORIZON

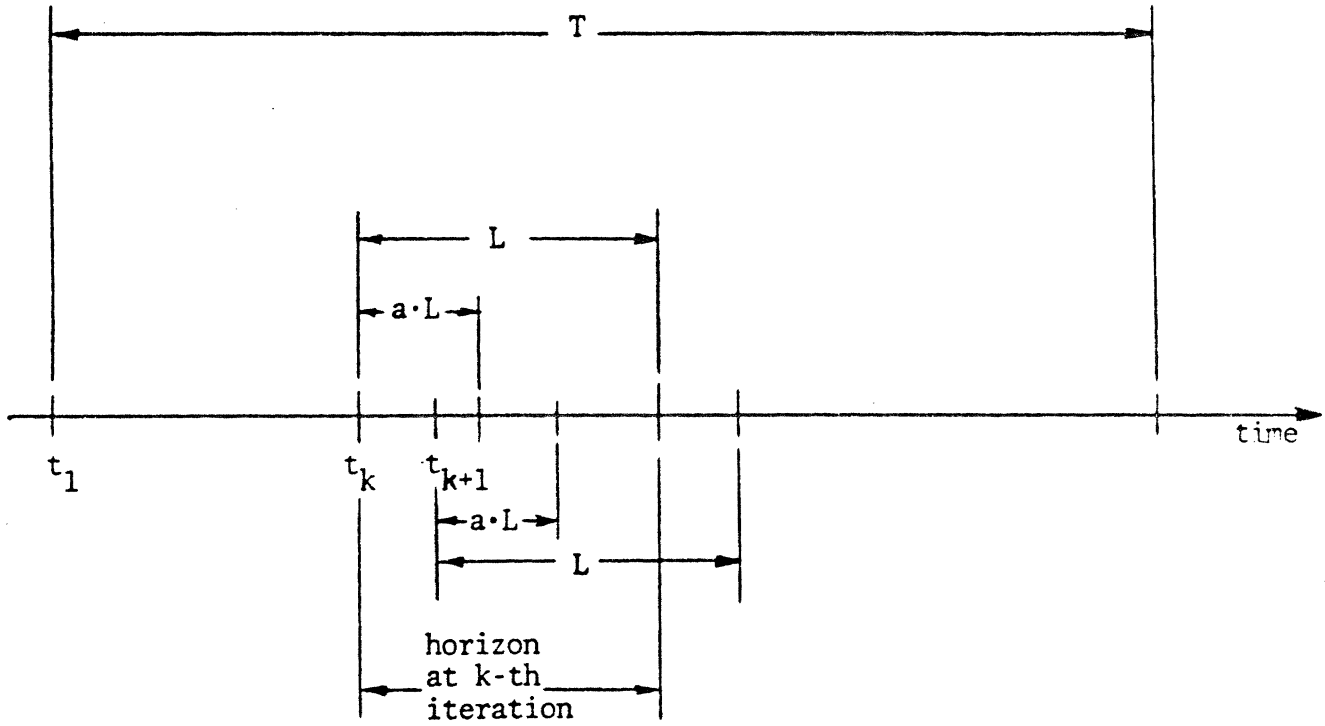


Figure 3.2

rolling horizon is a decomposition of the problem by time. The overall scheduling horizon T (which can be of the order of 180 days for long-range problems) is subdivided into time horizons of shorter duration L (say, of the order of two weeks). Within these shorter time horizons, assignments are made first on a tentative basis, and only cargoes within the "front end" of L are considered for permanent assignment. Parameter a specifies what fraction of L cargo assignments may be considered to become permanent. Thus, if L=2 weeks and a=0.5, MORSS will "look" at two weeks of cargo data at a time, and at each iteration will assign only one week of cargoes.

There are several positive features in this "rolling horizon" approach : First, we place a lesser emphasis on the less reliable future information on cargo movements. This advantage is of particular importance in emergency situations where movement information more than several weeks in advance may be subject to a number of alterations. Our approach focuses each iteration on near-term events but also has a look-ahead capability so that future scheduled events are taken into account. In addition, the initial ship positions are modeled explicitly within our framework. For these reasons, we believe that a time-decomposition approach is appropriate.

A crucial aspect of our approach is that the future is taken into account through information contained in adjacent time intervals. In particular, most ship voyages are scheduled over two consecutive intervals. The rolling horizon concept enables this connection of successive time intervals by examining overlapping time horizons. Thus, t_{k+1} will be typically smaller than t_k +L (see Figure 3.2) (an exception would occur if there are no cargoes having EPT's between t_{k+1} and t_k+L).

In Step 0 (initialization), and since the overall problem is dynamic in nature, the value of the time horizon length T may not necessarily be known to the scheduler in advance. In this case, T is set at some arbitrarily large value.

We now describe our modeling of the of utilities for each eligible cargo/ship pair.

3.3 Assignment Utilities

Within each time frame, MORSS makes assignments of cargoes to ships while taking into account assignments made in previous periods. At the k^{th} iteration, MORSS has already assigned cargoes leaving prior to t_k. In addition, some previously assigned cargoes may have pickup and/or delivery times scheduled after t_k. Because any subsequent assignment could cause changes in the anticipated delivery times of previously assigned cargoes, the utility of a proposed assignment takes into account both the projected delivery time of the assigned cargo and its effect on the delivery times of previously scheduled cargoes.

In addition, we will not assign too many ships to cargoes in the k^{th} scheduling horizon so as to be able to satisfy requirements for pickups in the subsequent two time horizons. In addition we model the delays caused by queueing at ports. (Our current model of queueing delays is a simple nonlinear estimate. We expect to refine our model in the future.)

These considerations lead us to conclude that the utility, or value of a proposed new cargo/ship assignment must depend on the following factors:

(1) the assignment's effect on the delivery time of the cargo to be considered for assignment;

(2) the assignment's effect on the delivery time of previously assigned cargoes;

and

(3) the assignment's effect on the system's ability to handle future cargo movement requirements:

(a) use of ship resources over the entire scheduling horizon;

(b) use of port resources over the entire scheduling horizon.

We now discuss each of these factors in detail:

3.3.1 Delivery Time Utility (For a Proposed Assignment)

In calculating this utility, and as we already mentioned in Chapter 2, we consider both the Earliest Pickup Time, (EPT), and the Earliest Delivery Time (EDT), to be "hard" constraints, that is, these constraints cannot be violated. We point out, however, that MORSS could be easily modified to handle "soft" EPT's and/or EDT's. This modification could be incorporated via changes to the method of calculating utilities, and would also increase the number of feasible schedules.

Although the EDT and EPT constraints are hard, we treat the latest delivery (LDT) LDT requirements as soft. To illustrate why, we consider a specified cargo movement requirement such that a combination of EPT, EDT, and LDT times is demonstrably infeasible. Specifically, in MSC-supplied data (see also Chapter 4) we have found a number of demonstrably infeasible cargo movements arising from the following situations: (1) the fastest ship of a given type is too slow to transport a cargo within the specified EPT-LDT interval; (2) only one ship can be available (at a

specific time) to satisfy as many as (say) three cargo movement requirements: It may pick up and deliver any two of the three cargoes within their specified EPT-LDT intervals, but for a variety of reasons it may not be able to do so for all three; (3) cargo movement requirements greatly exceed total available fleet capacity; and finally (4) port throughput limitations cause queuing delays at POE's and POD's. In addition, other factors such as ship-port restrictions, delays and medium-term saturation of shipping resources lead to LDT infeasibility for sets of cargoes. For these reasons we assume that the LDT constraints are "soft", that is, may be violated if necessary. We can then incorporate penalties for LDT violations into our objection function.

Given that latest delivery times are negotiable, we ask the question: How does the "value" (or "goodness") of a delivered cargo change as a function of delivery time? After extensive discussions with MSC personnel, we decided to answer this question as follows: First of all, each cargo c has a maximum possibile utility, say u(c). If the cargo is delivered early or on time (that is, if it arrives between its EDT and its LDT) its corresponding value is the maximum u(c). If it is only a few days late (say 1 or 2 days after its LDT) we assume its value is lower than u(c),but very close to maximum. If it is more than a few days late we assume its value decreases rapidly with delivery time delay until it reaches a minimum value, at some time LDT + t_0. We assume that delivery beyond LDT + t_0 does not substantially change the value of the delivered cargo, with that value remaining at its minimum. Parameter t_0 is user-input for each cargo, and is cargo-dependent. We have set t_0 = 14 days for the initial calibration of MORSS. By this assumption, a cargo delivered 2 weeks after its LDT is

worth about the same as if it were delivered 4 weeks after its LDT.

There are several possible functional forms for the delivery time utility of a cargo, all of which fit the above assumptions well. We have chosen a bell-shaped function because it is smooth and continuous and also matches our intuition better. It is also flexible due to four free parameters. A mathematical formulation for a bell-shaped function follows.

Let

U_c = Delivery time utility

V_{min} = Minimum utility (for very late cargo)

V_{max} = Maximum utility (for on time cargo)

t Tardiness of cargo (Arrival time - LDT if > 0, zero otherwise

t_0 = Time for which $U_c = V_{min} + 0.1(V_{max}-V_{min})$

b = Nonnegative exponent

Then

$$U_c = V_{min} + (V_{max} - V_{min})e^{-2(t/t_0)^b} \qquad (3.1)$$

The four parameters V_{min}, V_{max}, t_0 and b are user-inputs, and, in general, cargo-dependent (particularly V_{min} and V_{max}).

Graphs of U_c for several values of b are shown in Figure 3.3. Note that the case b = 2 corresponds to a shape similar to the curve for the Gaussian Probability Density Function.

3.3.2 A Proposed Assignment's Effect on Previously Assigned Cargoes

In our analysis of the effect of a proposed assignment on other previously assigned cargoes, we assume that, given a cargo's delivery time, that cargo's utility is independent of the delivery time of any other cargo. Specifically, for each cargo j, we assume that V_{max}, V_{min}, t_0 and

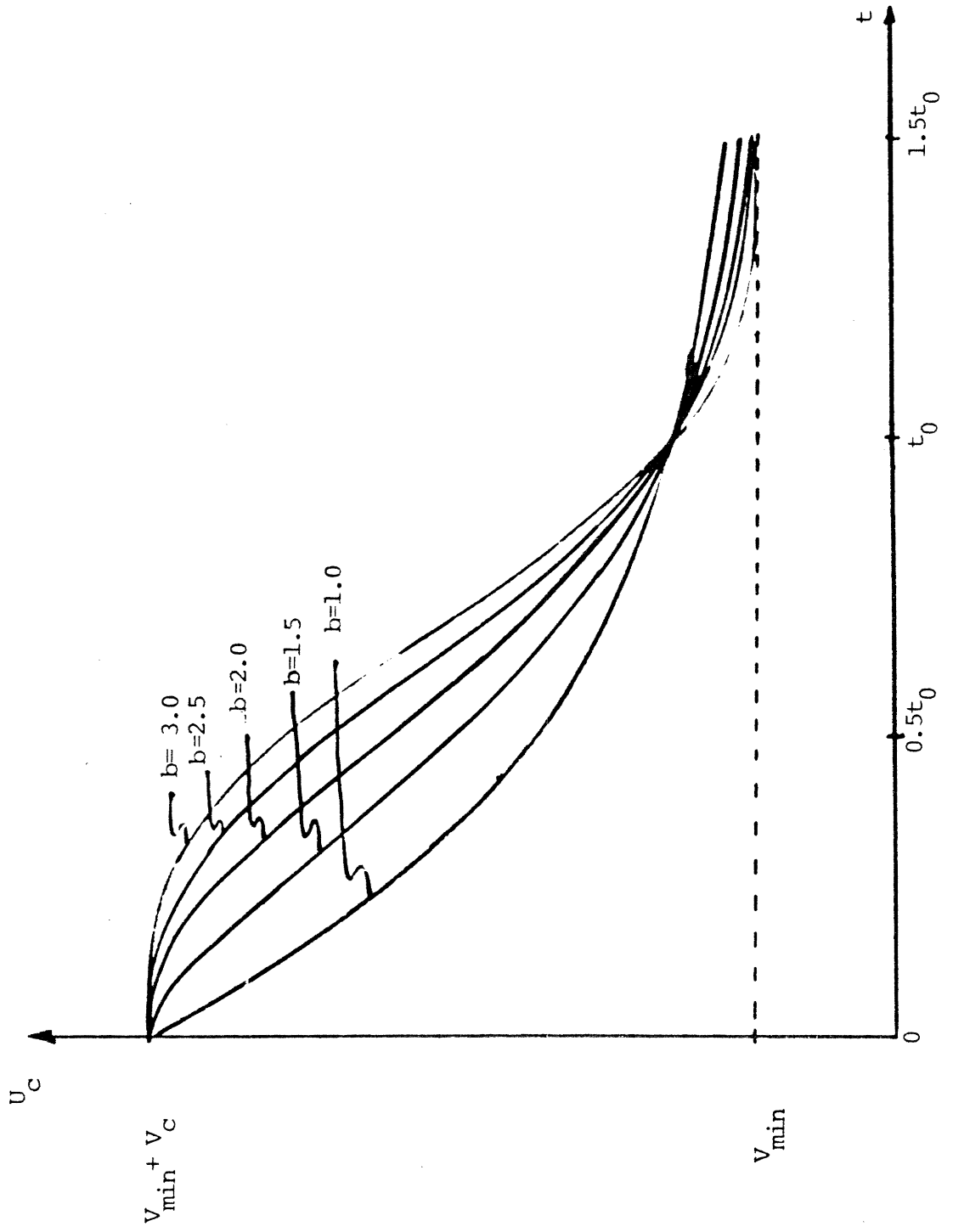

DELIVERY TIME UTILITY
U_c
$V_{min} + V_c$
V_{min}
b= 3.0
b=2.5
b=2.0
b=1.5
b=1.0
0
$0.5t_0$
t_0
$1.5t_0$
t

Figure 3.3

b are particular to cargo j; that is, they are independent of the delivery times of all other cargoes.

This assumption is reasonable given we have assumed no precedence constraints on the deliveries of cargoes (see Chapter 2). In the real world, exceptions to this rule may occur. For instance, delivering a particular cargo on time might be worthless (or even might be undesirable) if some other cargo has not been previously delivered. If such constraints do exist, MORSS handles them by user intervention and not internally.

Because of the independence assumption, the marginal effect of a proposed cargo/ship assignment on the delivery time utilities of other cargoes would be equal to the change in total delivery time utility of all previously assigned but yet undelivered cargoes (scheduled to be delivered by the ship in question only,) that occurs because of the addition to the proposed new cargo on that ship. In other words, the effect of a proposed assignment on a set of known assignments is the difference in the total delivery time utilities, with and without the proposed assignment. More rigorously, let

ΔU_D = effect of a proposed assignment on the utility of cargoes 1,...,n (already assigned to same ship)

U_j = utility of cargo j in original schedule (without new assignment)

U_j' = utility of cargo j in schedule which includes pickup and delivery of proposed assignment.

Then

$$\Delta U_D = \sum_{j=1}^{n} (U_j' - U_j) \qquad (3.2)$$

3.3.3 A Proposed Assignment's Effect on the System's Ship Resources

The primary motivation for this utility component is that ship resources are limited. There are a limited number of ships. Because of this limitation we wish ships to sail as fully loaded as possible. Thus ship utilization is important in assessing the goodness of a potential assignment.

It is rarely possible, however, to achieve 100% utilization on any leg of a ship's journey. Indeed, many ships often return (to pick up additional cargo) on ballast, which significantly decreases the average utilization. In addition, a high utilization may be impossible for a given problem. This might be because of the EPT/EDT/LDT structure of the problem. For instance, in MSC-supplied data we have seen that the assignment of a cargo to a ship severely limits that ship's ability to pick up and deliver other cargoes on time.

Looking at the use of ship resources in another way, and comparing two potential cargo assignments for a ship, (everything else being equal), we will tend to prefer the one which gives the ship the most flexibility (in terms of available slack in that ship's schedule) in carrying additional cargoes. Thus, "schedule flexibility" is also important in assessing the goodness of a potential assignment.

The value of "ship utilization" and of "schedule flexibility" are related in the following way: In terms of ship resources, the optimal condition is for the ship to be full. In this case schedule flexibility is unimportant since the ship can pick up no additional cargoes. (We assume

most trips are from an area of POE's to an area of POD's with no - or few - POD's in between). Here we have maximum utilization of shipping resources. The worst situation is for a ship to be empty (or nearly so), and, to have no schedule flexibility. Here the ship is essentially worthless, since it is both almost empty and has little flexibility in its schedule to pick up new cargo. This is the case where the ship is deadheading and on a tight schedule. Here we have zero use (i.e. waste) of shipping resources. Intermediate between these extremes is the situation where the ship is empty and has maximum schedule flexibility. This is the case where the ship has no future deliveries scheduled yet. Here no shipping resources are being used - the ship is empty - yet shipping resources are available because of the schedule flexibility. In this case we give a low, but intermediate value for utilization of ship resources, or ship utility.

These combinations are summarized in Table 3.1.

Ship Condition	Schedule Flexibility	Ship Utility
Full	High	Maximum
Full	Low	Maximum
Empty	High	Low
Empty	Low	Minimum

Table 3.1

For intermediate values of ship utilization and of schedule flexibility, ship utility is an increasing function of each component.

There are several functional forms which fit the above characterization of ship utility. Again we chose a bell-shaped model for MORSS for two reasons. (1) It more closely reflects our belief that

utility should change more rapidly at the intermediate ranges of the components than near the extreme values, and (2) its specific form is flexible due to five free parameters.

A mathematical formulation for such a bell-shaped function follows:

Let

U_s = Ship utility at a stop.

V_s = Maximum value of ship utility

R = Residual capacity of ship after the stop

C = hip capacity

F = Schedule flexibility (slack in schedule averaged over-all future stops)

L = Scheduling horizon length

c,d = Non-negative user inputs

f = User-input $0 < f < 1$

Then

$$U_s = V_s e^{-2(R/C)^c} (1 - f(F/L))^d \qquad (3.3)$$

A graph of this two-dimensional surface appears in Figure 3.4.

Note that the value of this utility is different at each stop on a ship's schedule. For any given stop, its value depends on the fractional ship utilization immediately after the stop (i.e. between the stop and the following stop), and on schedule flexibility averaged over all future stops. As a result, the value of ship utility for a proposed assignment must be a combination of ship utility at the pickup and at the delivery.

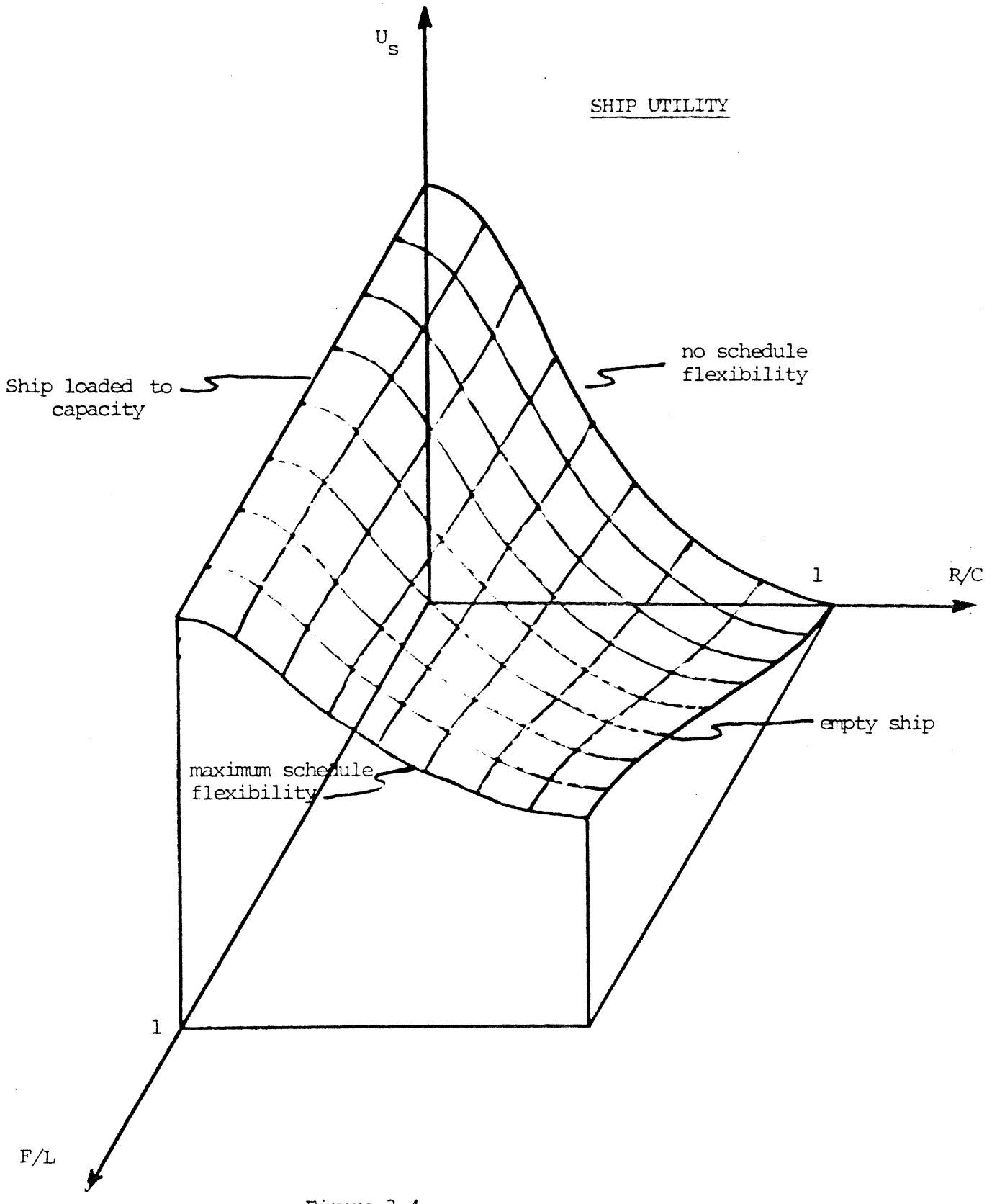

U_s
SHIP UTILITY
Ship loaded to capacity
no schedule flexibility
1
R/C
empty ship
maximum schedule flexibility
1
F/L

Figure 3.4

We have chosen to add these two utilities directly, so that

$$U_s = U_s(\text{pickup}) + U_s(\text{delivery}) \qquad (3.4)$$

where U_s = total ship utility for an assignment.

Our motivation for this choice is that is computationally simple and achieves our desired objective of encouraging assignments which utilize fewer ship resources.

3.3.4 A Proposed Assignment's Effect on the System's Port Resources

The motivation of this utility component is that port throughput capacity is limited, and queueing can cause enormous delays and waste over the long run, because of idle ships. Because of this limitation we wish to discourage additional ships from stopping at ports when they are near or over throughput capacity. In this case we wish to encourage a ship which has already been assigned to a port for another task, to handle the pickup or delivery of other cargoes in that port within the same time frame. Our method is the following. We divide the overall scheduling horizon into a series of "congestion periods" of length p. (We chose p=3 for the initial calibration runs). We then keep track of the number of stops at each port for each of the congestion periods.

The congestion level for a period is defined as the ratio of the number of scheduled stops in the period divided by the total throughput capacity, measured in number of ships, of the period. Since these congestion levels depend on the number of stops scheduled so far, and these numbers change with each new set of assignments, MORSS updates them at each iteration.

Port congestion and the value of port utility for a proposed cargo movement (pickup or delivery) are related as follows. The optimal condition is when no additional stop is required to fulfill the movement. A ship already assigned to the port can do the job. Here no additional port resources are used, so the value of port utility is maximum. If, however, an additional ship entry into the port is required to fulfill the task, then the situation is more complex and will in all cases, exhibit a lower utility. At higher levels of port congestion, available port resources are scarcer, so their value is higher. Similarly, at lower levels of congestion, port resources are abundant, hence cheaper to use. Consequently port utility is a decreasing function of congestion.

There are several functional forms which fit this description of port utility. As before, the bell-shaped curve is more attractive because it is smooth and is flexible because of four free parameters. A sample graph is shown in Figure 3.5.

A mathematical formulation for the bell-shaped curve is given below, for the value of port utility at a stop.

$$U_{ps} = \begin{cases} W_p & \text{If no additional stop is required} \\ V_p e^{-2(mN/P)^1} & \text{If an additional stop is required.} \end{cases} \quad (3.5)$$

where

U_{ps} = Port utility for the stop

W_p = Maximum value of port utility if no additional stop is required

V_p = Maximum value of port utility if additional stop required

N = Number of stops in port during congestion period in question

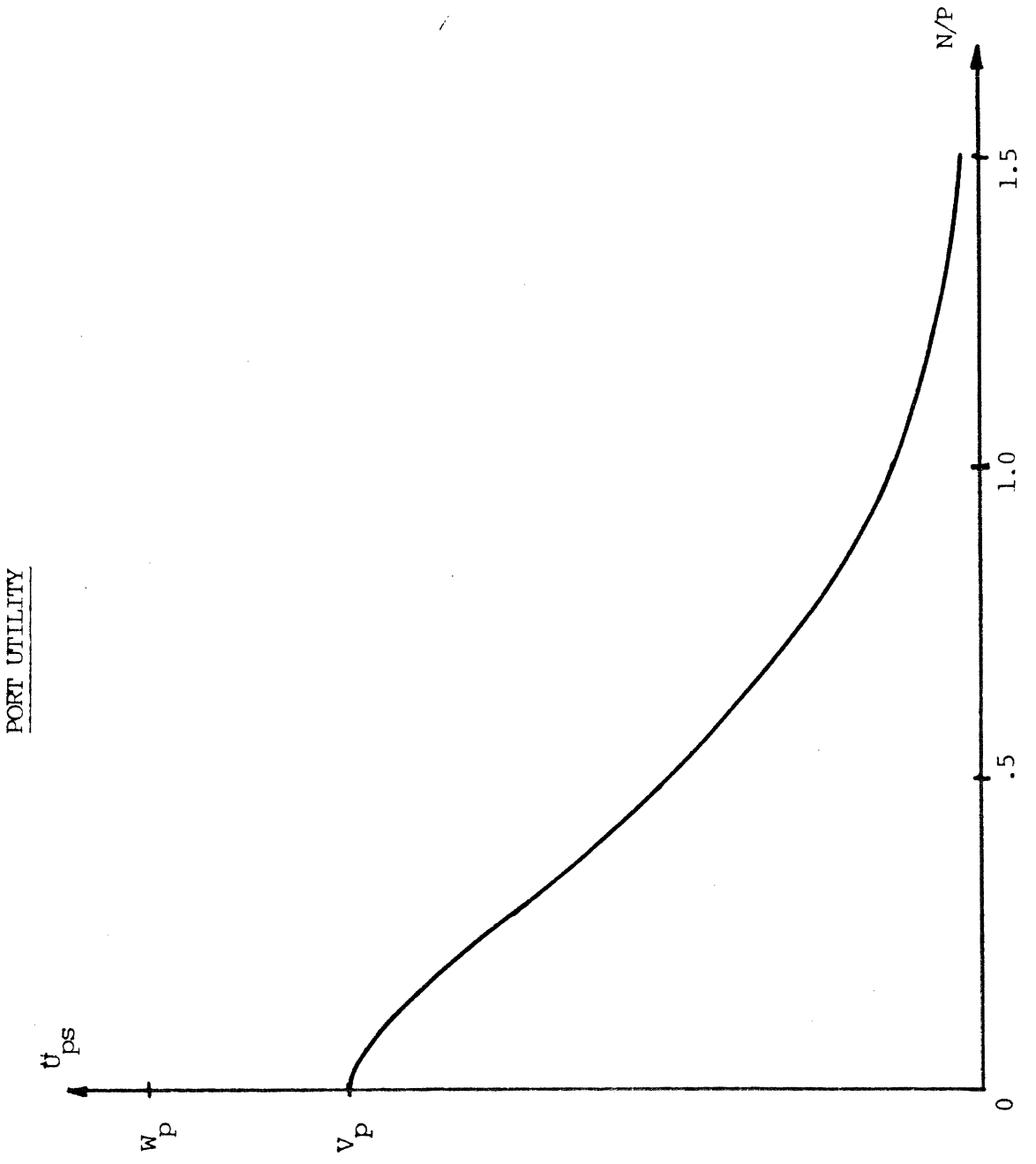


Figure 3.5

P = Throughput capacity of port, per congestion period

m,l = Nonnegative constants.

Since this utility is different for each stop on a ship's schedule, the total value of port utility for a proposed assignment must be a combination of the port utilities at the pickup and at the delivery. We have chosen to add these two utilities directly so that

$$U_p = U_{ps} \text{ (pickup)} + U_{ps} \text{ (delivery)}. \qquad (3.6)$$

where U_p = total port utility for the assignment.

Our motivation for this approach is that it achieves our primary objective: encouraging assignments which utilize fewer port resources, while being computationally simple. This functional form is but a first approximation of the queueing process at a port, a process which in itself merits further investigation (see also Chapter 5).

3.3.5 Total Assignment Utility

Our previous discussion has shown that the utility of a proposed assignment depends on four factors:

(1) its own delivery time

(2) its effect on other cargoes' delivery time,

(3) its use of system ship resources, and

(4) its use of (system) port resources.

For this initial phase of the project, we have expressed the total assignment utility as a weighted sum of four utility components. Mathematically,

$$U_t = U_c + \Delta U_D + U_s + U_p \qquad (3.7)$$

where

U_t = Total utility of an assignment

U_c = Delivery time utility for the assignment

ΔU_D = Effect of the assignment another cargoes U_c's

U_s = Ship utility for the assignment

U_p = Port utility for the assignment.

Note that the weighting factors for the component utilities are implicit in their calculations, because of the terms V_{min}, V_{max}, U_s, W_p, and U_p. A major part of the (initial) calibration of the model will be determining appropriate relative values for these weighting factors.

We now turn to the question of how SCHEDULE determines the assignment utility for a given ship-cargo pair. SCHEDULE does this in several steps. First, an incompatibility may exist if the ship type and cargo types do not match or if the port facilities at POE or POD cannot handle the ship because of draft, beam, heavy lift, etc. constraints. In this case the assignment utility is given a large negative value. Then SCHEDULE examines a set of possible ways to insert the cargo's pickup and delivery into the ship's existing schedule. For each of these insertion possibilities, SCHEDULE computes a utility value. If every insertion possiblity yields a net decrease in overall delivery time utility, i.e if $U_c + \Delta U_D < 0$ for all insertion possibilites, then the assignment utility is given a negative value. Otherwise from those insertion possibilities which realize a net increase in overall delivery time, SCHEDULE chooses the one with the maximum overall utility. The utility of the assignment is then set equal to the utility of this insertion possibility.

3.4 Solving the Assignment Problem

As we have noted previously, the main source of complexity in the Sealift problem is common to a broad class of scheduling problems. The extreme complexity of the problem stems from the non-linear interactions between utilities of different ship/cargo assignments. The utility of any cargo-ship assignment is directly dependent upon the assignment of other cargoes to that ship. It is also indirectly dependent on queueing at ports-of-call, caused by cargo assignments to other ships. As previously discussed, our approach is to decompose this problem by time into smaller problems, one per scheduling horizon. This decomposition reduces the complexity of the problem by several orders of magnitude, but does not bypass it entirely. The remaining complexity may be addressed by the question: On what basis do we simultaneously assign cargoes to ships when the assignment utilities' strongly depend on how ships schedules are modified by the assignments themselves? There are several possibilities.

One approach is to simultaneously assign all cargoes within the scheduling horizon, using utilities calculated from previous schedules. This has the advantage of simplicity and speed, but it ignores cargo interactions. We have thus rejected this approach.

A second approach is to assign cargoes one at a time, starting with the one with the highest assignment utility, or the one with the earliest EPT, and then recomputing utilities of all other eligible cargoes based on that assignment. This approach has the advantage of taking into account bad interactions in a myopic way. It fails to capture the favorable interactions between compatible sets of cargoes. It is also computationally slow. A modification of the second approach is to assign one cargo per ship and then recompute utilities at each iteration. This

forces ships to have the same number of cargoes, and fails to capture favorable interactions.

A third approach is to enumerate all possibilities. This method is computationally too expensive. There are of course many other modifications and types of approaches. From the three described above, we may, however, deduce principles from which to select an appropriate heuristic. We wish at the same time and with a reasonable computational effort, to: (1) capture as many favorable interactions as possible, and (2) eliminate bad interactions between assignments.

It should be clear that we are in a tradeoff situation with respect to these goals. We have therefore chosen a heuristic solution methodology which is a compromise between achieving the two goals. The heuristic allows multiple simultaneous assignments for each ship, via an optimally solved transportation problem. It assigns cargoes within each scheduling horizon, rejecting assignments which have strong negative interactions. Thus the algorithm maintains the computational speed of simultaneous assignments and the "goodness" of a solution which takes interactions into account. The rest of this section contains a detailed description of the heuristic (additional details on its computer implementation are in Appendix B).

Once assignment utilities are calculated for all eligible ship/cargo pairs, they are used to create a transportation network, (see Figure 3.6). The network is bipartite. Ships are sources, and eligible cargoes are sinks. To speed up the algorithm and to avoid excessive non-linear interactions ships are given a user-input integral supply of S (we chose S=4 for the initial calibration runs). S is the maximum number

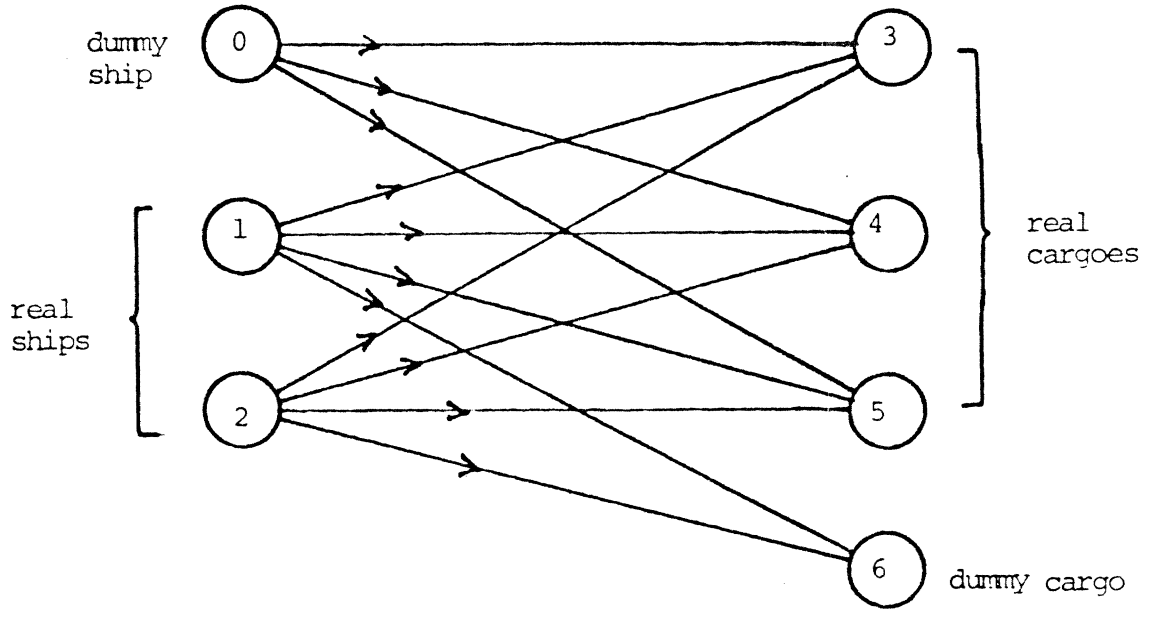


Figure 3.6

of simultaneous cargo assignments each ship may receive at any individual iteration (scheduling horizon). Cargoes are given a demand of 1 to reflect the fact that it must be assigned to a ship. In the network all compatible ship-cargo pairs are connected by directed arcs (from ships to cargoes) whose costs are the negatives of the corresponding assignment utilities. In addition, for bookkeeping purposes we create a dummy ship and a dummy cargo.

The dummy ship has arcs to all real cargoes, each with "high" cost. It is given an infinite supply, so it may take as many cargoes as necessary. Its function is to take any cargoes which are not assigned to other ships for any reason (incompatibility, limited shipping resources, etc). The dummy cargo serves the following internal bookkeeping function: It balances demand in the case where there is more ship supply than cargoes. To accomplish this, it has one arc from each real ship, each with a "high" cost. When the network is complete it is sent to subroutine FLOSUB. This routine finds an optimal solution to the transportation problem. It maximizes the net overall utility of assignments in the scheduling horizon, while assigning each cargo (dummy and real) to one ship (dummy or real). The routine FLOSUB uses an especially fast method written by Orlin (1983a, 1983b). The mechanics are described in more detail in Appendix B. The routine outputs a list of tentative feasible ship-cargo assignments. How these are dealt with is described below.

3.5 Permanent Assignments

At each iteration MORSS divides the tentative assignments in the interval $(t_k, t_k + L)$ returned by FLOSUB into three categories: (1) new assignments which are unique to a ship; (2) new assignments which are not

unique to a ship (that is, there is more than one tentative assignment to the ship); and (3) assignments to the dummy ship. These are treated as follows.

Assignments which are unique to a ship are made permanent. There are no non-linear interactions with other cargoes assigned to the ship.

Assignments which are not unique to a ship are further divided by ship. On each ship, the tentative assignment with the earliest EPT is made permanent. The remaining tenative assignments on the ship are looked at in order of increasing EPT. One by one their assignment utility is recalculated based on the ship schedule updated by newly made permanent assignments. If the net decrease in utility is less than a user-input critical value (percentage and/or absolute measure of change) the assignment is made permanent. If the decrease in utility is more than the critical value, then the cargo is returned to the pool of unassigned cargoes.

Assignments to the dummy ship are also returned to the pool of unassigned cargoes. Assignments whose EPT falls in the interval ($t_k + aL$, $t_k + L$) are not assigned permanently. Their function, as discussed previously, is to link successive scheduling horizons. They are all returned to the list of unassigned cargoes.

When a permanent assignment is made, it may happen that the cargo is too large for available ship capacity. In this case the cargo is split. As much as possible goes on the ship, and the remaining amount is returned to the pool of unassigned cargoes for the next iteration.

In order to prevent endless cycling caused by an infeasible cargo, we limit the number of times a cargo may be returned to the pool of

unassigned cargoes. Once it has exceeded that number (we chose 4, initially), it is added to the list of infeasible cargo movement requirements. It is therefore possible that MORSS will leave some cargoes unassigned. This is sometimes the most desirable outcome. Numerous examples of infeasibilities exist in the MSC - supplied database, as already discussed in section 3.3.1. However, MORSS leaves cargoes unassigned only when the available alternatives have greater utility. Since utility decreases with delivery time, unassigned cargoes are possible.

3.6 Seed Assignments

Another concept central to MORSS is the concept of "seed" assignments. This is a one-to-one assignment of some cargoes (seed cargoes) to some ships early on in the scheduling process so that a good starting solution is obtained. Such a solution serves as a "skeleton" for the final schedule, which gradually evolves from the seed schedule as subsequent assignments are made at future iterations. As in other assignments, seed selection is performed by solving an assignment problem whose objective function is the maximization of the total "utility" of the assignment. For each eligible seed cargo/ship pair, we again compute the "utility" of the corresponding pair. The main differences between seed assignments and subsequent assignments are (a) seed assignments are made on a one-to-one basis whereas in subsequent assignments more than one cargo can be simultaneously assigned to a ship, and (b) subsequent assignments take into account assignments already made at prior iterations while this is not applicable in seed assignments.

At the present stage of our research, seed assignments are selected

by the MORSS user in an interactive fashion. We anticipate developing a utility-based assignment approach, as discussed above, in the next phase of our research.

CHAPTER 4

COMPUTATIONAL EXPERIENCE

4.1 Introduction

In this chapter we describe our computational experience with the MORSS algorithm as it relates to data supplied to us by the MSC. We should mention at the outset that the data we have used so far have been "sanitized" by the MSC so as to avoid the disclosure of sensitive information. Such a "sanitization" process has been necessary given the unclassified nature of the MIT project (or, of any MIT research project for that matter). This chapter begins with a brief description of the MSC database (Section 4.2) and proceeds with a presentation of gross feasibility analyses that have been made with respect to the data (Section 4.3). Section 4.4 outlines our initial experience with the MORSS algorithm, performed on a small subset of the MSC database, including an interpretation of the results.

4.1 The MSC Database

The MSC database includes information on 505 cargoes, 232 ships and 26 ports.

Each of the cargoes is either a single item or a collection of items that have a distinct POE and a distinct POD. In addition, each cargo has a preferred ship type, an EPT, an EDT and an LDT. Also known is that cargo's Short Tons (STONS), its Measurement Tons (MTONS) and its Deck Area (SQFT). The 505 cargoes are classified into 8 categories, according to preferred ship type: These are breakbulk (193 cargoes), seatrain (13 cargoes), RO/RO (151 cargoes), self-sustaining container (54 cargoes), tanker (45 cargoes), and barge carrier (25 cargoes), while 4 cargoes are of

an unspecified ship preference.

Ships are classified into 3 fleet types and 6 ship types: The fleet type codes are MSC-controlled (38 ships), Ready Reserve (38 ships) and Sealift Readiness Program (156 ships). The ship type codes are breakbulk (100 ships), seatrain (5 ships), RO/RO (5 ships), self-sustaining container (9 ships), tanker (25 ships) and barge carrier (18 ships). Information on ships includes capacity (weight/volume/deck area), draft, speed, cargo loading and unloading rates, as well as initial location.

Finally, the 26 ports of the MSC database are classified as follows: 12 are POE's, 5 are POD's, and the remaining 9 are both POE's and POD's. All ports of the database are located in the United States, the Panama Canal Zone and the Pacific. Information on ports includes the throughput characteristics of their various berths and terminals. All inter-port distances are also known.

Various statistics of the MSC database (such as range of cargo sizes, general cargo movement patterns, etc), have been reported in Chapter 5 of Jeng (1984). Here we focus on analyses that can be quickly performed to ascertain the gross feasibility ofa particular problem instance. The following section provides the rationale that has been used on that score.

4.3 Gross Feasibility Analysis

Given a particular "problem instance" (that is, a set of prescribed inputs for the MSC operational problem), we have assumed that the scheduler would like to know early on in the decision-making process to what extent this particular instance is feasible. That is, the scheduler would like to obtain a preliminary idea regarding whether available resources are enough to satisfy the prescribed cargo movement requirements. If the opposite

turns out to be the case, it would make little sense to proceed with a detailed scheduling run, because most cargoes would be delivered late. Instead, it would then make sense to relay infeasibility information immediately to the chain of command "upstream", so that either the cargo movement requirements are modified, or additional resources are made available, or some other measure is taken to alleviate this problem. Thus, the MIT team considered important that a set of simple feasibility tests be developed, so that a "quick-and-dirty" picture of the feasibility (or lack thereof) of a particular instance is established.

Gross feasibility analysis can be performed at various levels of detail. At the simplest level, one can check whether the prescribed cargo delivery time requirements alone are reasonable or unreasonable by performing a screening test that will be described below. Such a test is always an optimistic estimator of feasibility, in the sense that any problem instance identified by the test as "bad" is always infeasible, whereas an instance not identified as "bad" is not necessarily feasible. Such an instance (that is, one that has "passed" this first feasibility screening test) can be further tested for feasibility by more sophisticated tests (see later description), and, ultimately, be fed as input to MORSS for detailed scheduling. Of course, if a problem instance fails to pass this first screening test, it is "rejected" and sent back to the chain of command upstream for further action.

As said before, the simplest screening test that can be performed concerns cargo information only. For a particular cargo, define SLACK(V) = (LDT-EPT)-(Direct transit time between that cargo's POE and POD if ship speed is V knots). This represents the maximum time slack between that

cargo's actual delivery time and its LDT, under the optimistic assumption that the cargo leaves its POE immediately at its EPT and travels directly to its POD.

Define also for that cargo SSPEED as the ratio of its POE/POD distance divided by (LDT - EDT) and expressed in knots. This optimistically represents the minimum ship speed required if the cargo leaves its POE promptly, is delivered to its POD at its LDT, and travels directly.

Figure 4.1 is a histogram of SLACK(15) for all 193 cargoes belonging to the breakbulk category (that is the largest cargo class in the MSC database by number). Since many of those cargoes have negative SLACK(V), and since this statistic is only an optimistic representation of the actual slack time, the histogram shows that delivering all of these cargoes on time is virtually impossible, irrespective of both actual ship resources and scheduling strategy.

A similar conclusion can be drawn for Figure 4.2, which is a histogram of SSPEED for the same cargo sample. Given it is rather unlikely to have breakbulk ships with speeds of more than 25 knots immediately available, we can conclude that meeting cargo deadlines for this problem instance is virtually impossible, whatever ships are available and whatever algorithm is used for the scheduling.

Similar observations were made in the MSC database on virtually all other cargo/ship categories. Put in another way, the MIT team discovered that the MSC-supplied database suffered from a widespread degree of gross infeasibility, at least as far as the possibility of meeting cargo due dates was concerned. From our discussions with MSC personnel, we

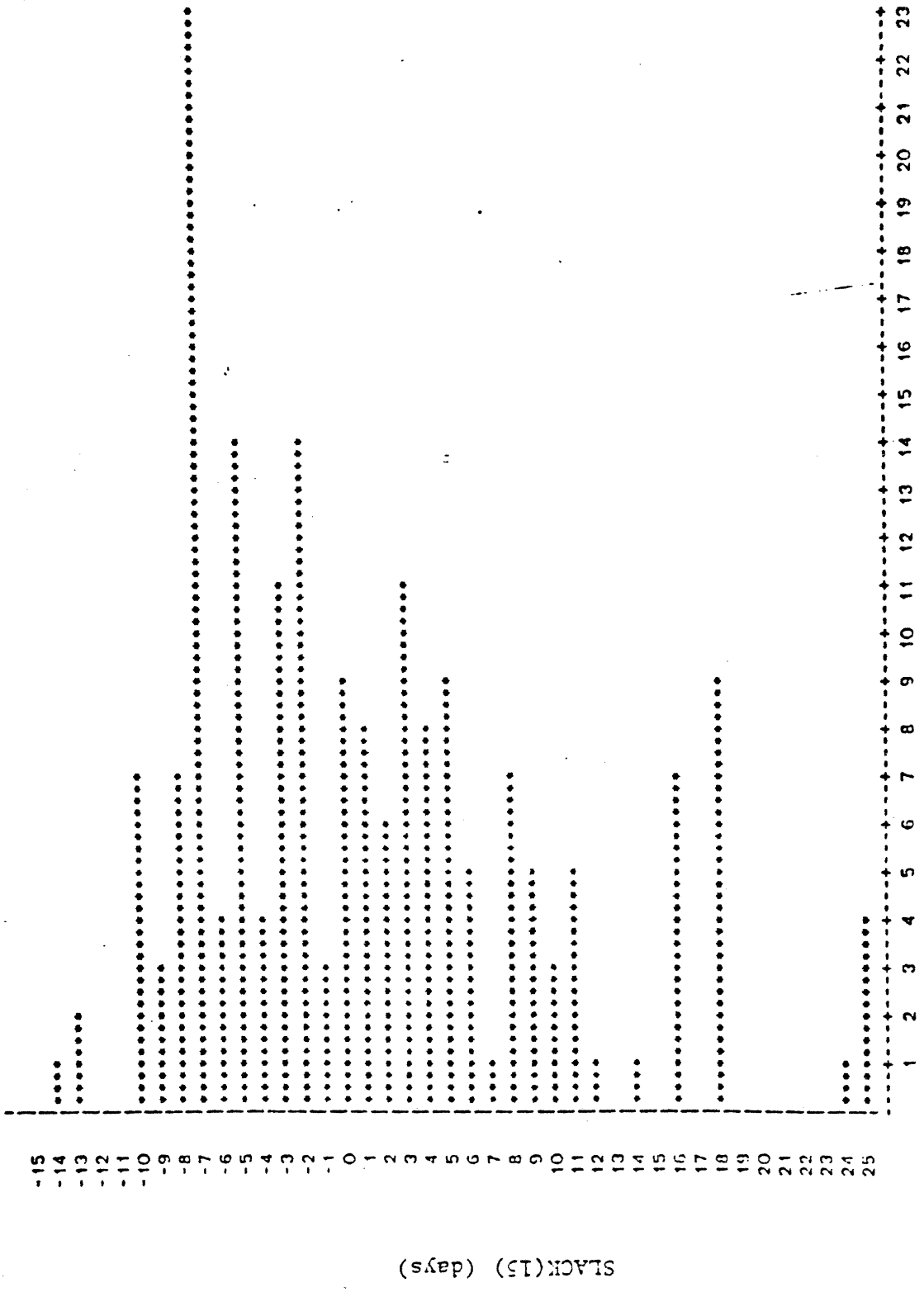


Figure 4.1 : Histogram of SLACK(15)

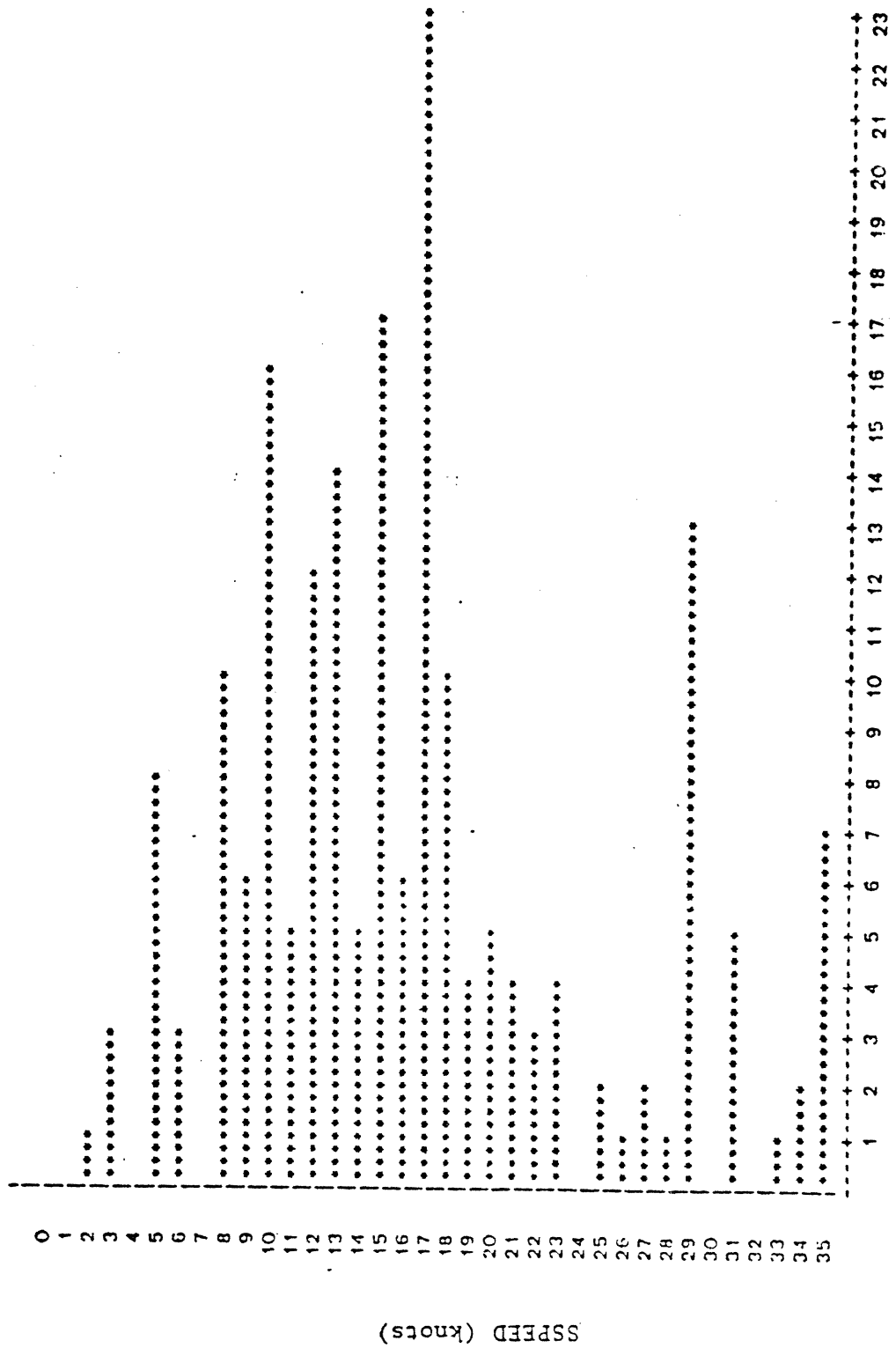


Figure 4.2: Histogram of SSPEED

understood that the infeasibility of the database could be attributed in part to the "sanitization" process that was used to convert the database from classified to unclassified. Other factors might be valid as well, such as the facts that in practice a cargo's EPT and LDT are sometimes (if not always) determined by two different (and sometimes unrelated) decision processes. As a result, a cargo's LDT usually reflects neither that cargo's availability at its POE, nor its POE/POD distance (In fact, in the MSC database we even found that in some cases a cargo had an LDT which was earlier than its EPT - a clearly absurd requirement).

As a result of the above, the MIT team decided to use only smaller, more manageable, and closer-to-feasible parts of the MSC database to test the MORSS algorithm. This made sense since we did not want to begin the testing of MORSS with huge, grossly infeasible amounts of data. In the meanwhile, another database was prepared by the MSC and sent to MIT for further testing. This new database arrived at MIT rather late for inclusion in this report (August, 1985). This new database would be used in conjunction with further calibration and testing of MORSS (see also Chapter 5).

Before we describe our limited computational experience with MORSS, we conclude this section by outlining a second-level, more sophisticated gross feasibility test. Such a test could be used for data that have passed the first test outlined earlier.

By contrast to the previous test which completely ignores ship resource information, that is, is based only on an evaluation of "demand" (cargo-related/requirements, this second-level test takes also into account the "supply" side of the problem, that is, the ability of available ships

to meet that demand.

The second-level test formulates the feasibility problem as a "transportation" problem. It sets up a bipartite network very similar to the one described in Chapter 3, with supply node i (i=1,...,m) representing ship i, and demand node j(j=1,...,n) representing cargo j. Let a_i be the capacity of ship i, w_j be the size of cargo j and c_{ij} be the tardiness of delivering cargo j by ship i, assuming a direct and immediate service by ship i from the POE to the POD of cargo j. We then define as x_{ij} the capacity of ship i allocated to cargo j and solve the following transportation problem:

$$\text{minimize} \quad \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij}$$

$$\text{s.t.} \quad \sum_{j=1}^{n} x_{ij} \leq a_i \qquad (i = 1...,m)$$

$$\sum_{i=1}^{m} x_{ij} = w_j \qquad (j = 1,..,n)$$

$$x_{ij} \geq 0$$

It is clear that the optimal value of this problem is a lower bound on the actual total weighted tardiness (ton-days late) that would incur under any actual allocation of available ships to cargoes. Such a test is actually identical to the first iteration of SAI's transportation algorithm (SAI, 1982). Again, as in the previous case, this test is optimistic in the sense that it always identifies a "bad" instance but may not necessarily correctly proclaim one as "good". The latter, in a sense, can only be assessed after detailed scheduling has been attempted.

Given the above considerations regarding the "reasonableness" of the MSC-supplied database, the scope of any test runs of MORSS with that database became immediately limited. Nevertheless, in spite of the

infeasibilities of the database, we began testing the algorithm with subsets of the data. This is described in the following section.

4.4 MORSS Test Runs

We chose a subset of the MSC database for the initial runs of MORSS with certain goals in mind. For ease of analysis we wanted ships and cargoes to be of compatible types. We desired the problem size to be at once small enough to easily manage and comprehend, and large enough to contain the complexities of the MSC operational problem. Finally, we wanted a nontrivial problem, one which reflected to some extent the infeasibility of some of the time windows in the MSC database. In short, we sought for a representative database for the initial runs.

To meet these goals, we selected a set of twenty breakbulk cargoes. The LDT's of these cargoes ranged from day 8 to day 42. Their size ranged from 10 measurement tons to 264,414 measurement tons. The POE's included four Atlantic-side ports from New York to Galveston, three Pacific-Side ports from Los Angeles to Seattle, and four Pacific Ocean island locations. The POD's included the Philipines, Malaysia and Eastern Australia. We took relevant information for each of these ports (e.g. depth, width, interport distances) from the MSC database.

In analyzing the cargo data, we found that the roundtrip time of a ship was long in comparison to the range of the cargo LDT's. This precluded the possibility of multiple trips by the same ship. We therefore modified the EPT, EDT, LDT times to cover a 98-day span. This change allowed a relatively small number of ships to meet most of the prescribed cargo movement requirements, while completing as many as four (4) roundtrips in the Pacific theatre, and two (2) roundtrips from East Coast

POE's. Cargo characteristics are listed in Table 4.1.

We chose a fleet of seven (7) ships for the initial runs. Their speed ranged from 432 to 552 nautical miles per day. Their capacity ranged from 3,800 to 19,125 measurement tons, and from 6,000 to 7,500 square feet of deck space. For each ship, we took availability times at each port from the MSC-provided database, as well as other relevant ship characteristics, e.g. draft, beam, boom capacity, etc.

Our objective in the initial computational study was twofold. First, we wanted to verify that MORSS was operating correctly and in a manner consistent with its design. Secondly, we desired to begin the gross-scale calibration of the model, that is, empirically testing the effect of seed assignments, cargo size and time window arrangement on the quality of the overall solution (see also Chapter 5). Our procedure was also two-fold. First, we tested MORSS on increasingly larger problems until we were satisfied that the model was debugged. Then we began the gross-calibration study, which we now discuss.

The initial runs confirmed our intuition that seed assignments greatly influence the quality of the final schedule. Our strategy was to try a variety of seed assignments in order to gain a measure of the quantitative effects of this key factor. These assignments were made randomly at first, then varied to test various hypotheses about what constitutes "good" seed assignments. In particular, we attempted to improve the quality of the solution (e.g. decrease the number of undelivered cargoes) by modifying the assignment of seeds from one run to the next.

The complete man-machine session for the first run (Data Set A) is

Cargo #	POE	POD	EPT	EDT	LDT	Quantity (tons) A	B	C
1	Los Angeles	Philippines	5	14	20	10680	-	-
2	S. Japan	Malaysia	5	20	27	14830	-	4830
3	Norfolk	Philippines	11	10	30	4150	-	14150
4	Galveston	Philippines	12	14	35	690	6900	-
5	New York	Malaysia	18	28	40	10	1000	-
6	Charleston	Malaysia	25	25	47	30	3000	-
7	Norfolk	Philippines	25	5	53	195200	9520	-
8	Charleston	Malaysia	31	10	57	264140	26414	12641
9	Norfolk	Philippines	34	19	60	29030	-	12903
10	Hawaii	Philippines	35	9	65	7430	-	-
11	San Francisco	Philippines	45	13	69	190	1900	-
12	Wake	Philippines	45	45	73	1950	-	-
13	Norfolk	Philippines	50	26	78	30	3000	-
14	Charleston	Malaysia	55	30	80	16480	6480	-
15	Okinawa	Malaysia	55	4	84	24650	2465	-
16	Okinawa	Malaysia	57	6	87	51880	5188	-
17	Okinawa	Malaysia	60	2	90	4120	-	-
18	Japan	Malaysia	65	6	92	1910	-	-
19	S. Japan	Philippines	65	11	94	150	1500	-
20	New York	Australia	70	33	98	17550	7550	-

Table 4.1: Cargo Data Set (A, B, C are variants)

displayed in Table 4.2. These results show the effect of a gross infeasibility in the problem structure. Two cargoes (#7 and #8) are very large: Together they require 25 trips by the largest ship. The next largest three cargoes require 6 roundtrips. By contrast, the smallest five cargoes fill 3.0% of the smallest ship. This large disparity between ship size and cargo movement requirements allows only 11% of the total cargo weight to be delivered on time, while 35% of the total number of cargoes arrive on time. Because of this gross infeasibility, we modified the measurement tons of twelve of the cargoes to even out the spread. The revised numbers (Data Set B) appear in Table 1, column B. For this data set only two cargoes require multiple ship-trips. Summary statistics from runs 2,3, and 4 appear in Tables 4.3, 4.4 and 4.5 respectively (the complete man-machine sessions are omitted due to space limitations). These runs use seven ships and several seed assignment configurations. Some infeasibility remains since less than 50% of the tons are delivered on time, and at most 60% of the cargoes arrive on time. We decided to further modify the cargo sizes to make the problem more feasible, but first we tried to improve the ship utilization, which is less than 7% for these runs. (Note that in measuring ship utilization we also count ballast legs, in which utilization is zero. This means that if a schedule includes ballast legs and multiple rountrips, ship utilization should not be expected to be high - see also discussion at the end of this section). To this end we made three runs (summary statistics in tables 4.6, 4.7 and 4.8) with 5 ships in various seed configurations. As anticipated, ship utilization rose, to an average of 10.6%. The percentage of tons delivered on time dropped by nearly half, and the number of cargoes delivered on time

```
R; T=0.01/0.01 14:03:35
SCHED
EXECUTION BEGINS...
  8 SUPERCOMPLEXES
 DONE READING IN COMPLEX AND PORT DATA
 READ IN   20 CARGO RECORDS
 READ IN    7 SHIP RECORDS
 ENTER OUTPUT STREAM RATING, SCALE   0 TO  10
1
 OUTPUT SET AT     1

                *** SEEDS ***
 INPUT SHIP AND CARGO NO.
1 1
 INPUT SHIP AND CARGO NO.
2 2
 INPUT SHIP AND CARGO NO.
3 3
 INPUT SHIP AND CARGO NO.
4 4
 INPUT SHIP AND CARGO NO.
5 5
 INPUT SHIP AND CARGO NO.
6 6
 INPUT SHIP AND CARGO NO.
7 7
 INPUT SHIP AND CARGO NO.
0
 INPUT WIDTH ,LIMIT
99 20

**********************************************************************

SCHEDULE FOR SHIP NO.    1:
TYPE   MTON   SQFT   SPEED
   1  14125   6000     432

DATE  CMP NAME  P_D   QUANT   RMRTN SLK CRG EPT EDT LDT   MTON TYPE
  14    9 L.A.  P    10680    3445   0   1   5  14  20  10680    1
  29   21 PHIL  D    10680   14125   0   1   5  14  20  10680    1
  54    5 CHAR  P    14125       0   0   8 31  10  57 264140    1
  82   22 MALA  D    14125   14125   0   8 31  10  57 264140    1

**********************************************************************

SCHEDULE FOR SHIP NO.    2:
TYPE   MTON   SQFT   SPEED
   1  14467   6000     444

DATE  CMP NAME  P_D   QUANT   RMRTN SLK CRG EPT EDT LDT   MTON TYPE
  32   14 SJAP  P    14467       0   0   2   5  20  27  14830    1
  38   22 MALA  D    14467   14467   0   2   5  20  27  14830    1
  65   14 SJAP  P     1910   12557  21  18 65   6  92   1910    1
  71   22 MALA  D     1910   14467   0  18 65   6  92   1910    1
  88   11 SEAT  P      150   14317   0  19 65  11  94    150    1
```

Table 4.2(a): 7 Ships, Original Data Set (A)

101	21	PHIL	D	150	14467	0	19	65	11	94	150	1

****

SCHEDULE FOR SHIP NO. 3:

TYPE	MTON	SQFT	SPEED
1	14467	6000	444

DATE	CMP	NAME	P_D	QUANT	RMRTN	SLK	CRG	EPT	EDT	LDT	MTON	TYPE
11	4	NORF	P	4150	10317	1	3	11	10	30	4150	1
36	21	PHIL	D	4150	14467	0	3	11	10	30	4150	1
45	13	WAKE	P	1950	12517	3	12	45	45	73	1950	1
51	21	PHIL	D	1950	14467	0	12	45	45	73	1950	1
76	5	CHAR	P	14437	30	0	14	55	30	80	16480	1
77	4	NORF	P	30	0	0	13	50	26	78	30	1
102	21	PHIL	D	30	30	0	13	50	26	78	30	1
105	22	MALA	D	14437	14467	0	14	55	30	80	16480	1

****

SCHEDULE FOR SHIP NO. 4:

TYPE	MTON	SQFT	SPEED
1	14467	6000	444

DATE	CMP	NAME	P_D	QUANT	RMRTN	SLK	CRG	EPT	EDT	LDT	MTON	TYPE
14	8	GALV	P	690	13777	0	4	12	14	35	690	1
39	21	PHIL	D	690	14467	0	4	12	14	35	690	1
53	10	S.F.	P	190	14277	0	11	45	13	69	190	1
58	18	HAWA	P	7430	6847	0	10	35	9	65	7430	1
69	21	PHIL	D	7430	14277	0	10	35	9	65	7430	1
69	21	PHIL	D	190	14467	0	11	45	13	69	190	1
95	2	N.Y.	P	14467	0	0	20	70	33	98	17550	1
117	24	AUST	D	14467	14467	0	20	70	33	98	17550	1

****

SCHEDULE FOR SHIP NO. 5:

TYPE	MTON	SQFT	SPEED
1	13800	6000	432

DATE	CMP	NAME	P_D	QUANT	RMRTN	SLK	CRG	EPT	EDT	LDT	MTON	TYPE
25	4	NORF	P	13790	10	1	7	25	5	53	195200	1
26	2	N.Y.	P	10	0	0	5	18	28	40	10	1
52	21	PHIL	D	13790	13790	0	7	25	5	53	195200	1
55	22	MALA	D	10	13800	0	5	18	28	40	10	1
60	17	OKIN	P	13800	0	0	15	55	4	84	24650	1
65	22	MALA	D	13800	13800	0	15	55	4	84	24650	1
93	2	N.Y.	P	3083	10717	0	20	70	33	98	17550	1
115	24	AUST	D	3083	13800	0	20	70	33	98	17550	1

****

SCHEDULE FOR SHIP NO. 6:

TYPE	MTON	SQFT	SPEED
1	19125	7500	552

Table 4.2 (b)

DATE	CMP	NAME	P_D	QUANT	RMRTN	SLK	CRG	EPT	EDT	LDT	MTON	TYPE
25	5	CHAR	P	30	19095	11	6	25	25	47	30	1
47	22	MALA	D	30	19125	0	6	25	25	47	30	1
57	17	OKIN	P	4155	14970	6	16	57	6	87	51880	1
55	17	OKIN	P	10850	4120	2	15	55	4	84	24650	1
60	17	OKIN	P	4120	0	9	17	60	2	90	4120	1
64	22	MALA	D	4155	4155	0	16	57	6	87	51880	1
68	17	OKIN	P	4155	0	0	16	57	6	87	51880	1
72	22	MALA	D	10850	10850	0	15	55	4	84	24650	1
76	17	OKIN	P	10850	0	0	16	57	6	87	51880	1
80	22	MALA	D	4120	4120	0	17	60	2	90	4120	1
84	17	OKIN	P	4120	0	0	16	57	6	87	51880	1
88	22	MALA	D	4155	4155	0	16	57	6	87	51880	1
88	22	MALA	D	10850	15005	0	16	57	6	87	51880	1
88	22	MALA	D	4120	19125	0	16	57	6	87	51880	1

****

SCHEDULE FOR SHIP NO. 7:

TYPE	MTON	SQFT	SPEED
1	19125	7500	552

DATE	CMP	NAME	P_D	QUANT	RMRTN	SLK	CRG	EPT	EDT	LDT	MTON	TYPE
25	4	NORF	P	19125	0	11	7	25	5	53	195200	1
45	21	PHIL	D	19125	19125	0	7	25	5	53	195200	1
65	5	CHAR	P	2043	17082	0	14	55	30	80	16480	1
87	22	MALA	D	2043	19125	0	14	55	30	80	16480	1

****

UNSCHEDULED CARGOES:

CRG	POE	NAME	POD	NAME	EPT	EDT	LDT	MTON	TYPE
2	14	SJAP	22	MALA	5	20	27	14830	1
7	4	NORF	21	PHIL	25	5	53	195200	1
8	5	CHAR	22	MALA	31	10	57	26414 0	1
9	4	NORF	21	PHIL	34	19	60	29030	1
16	17	OKIN	22	MALA	57	6	87	51880	1

****

****

SUMMARY OF CARGO STATISTICS

MEAN CARGO TARDINESS	=	8.80
STD.DEV. CARGO TARDINESS	=	8.95
MEAN WEIGHTED CARGO TARDINESS	=	69466
STD.DEV. WEIGHTED CARGO TARDINESS	=	119910
PERCENT CARGOES DELIVERED ON-TIME	=	35.00 %
PERCENT CARGOES DELIVERED LATE	=	60.00 %
PERCENT COMPLETELY DELIVERED CARGOES	=	75.00 %
PERCENT PARTIALLY DELIVERED CARGOES	=	20.00 %
PERCENT UNDELIVERED CARGOES	=	5.00 %

Table 4.2(c)

```
PERCENT TONS DELIVERED ON-TIME        =       10.84 %
PERCENT TONS DELIVERED LATE           =       16.26 %
PERCENT TONS UNDELIVERED              =       72.90 %

**********************************************************

**********************************************************

            SUMMARY OF SHIP STATISTICS

MEAN ROUTE CIRCUITRY                  =        1.31
MEAN WEIGHTED ROUTE CIRCUITRY         =        1.24
MEAN SHIP UTILIZATION                 =       11.93 %
MEAN DISTANCE TRAVELED                =    34585
STD.DEV. OF DISTANCE TRAVELED         =     8002
MEAN TON-MILES                        =  2.21E+08
STD.DEV. OF TON-MILES                 =  1.42E+08
MEAN SHIPPING DISTANCE                =     8839
MEAN PORT DELAY PER STOP              =        1.250
STD.DEV. OF PORT DELAY PER STOP       =        5.013
MEAN PORT DELAY PER SHIP              =        9.286
STD.DEV. OF PORT DELAY PER SHIP       =       11.250
MEAN PERCENT OF TIME SPENT IN PORT    =       11.84 %
MEAN ROUTE TIME                       =       78.43
STD.DEV. OF ROUTE TIME                =       16.76
R; T=3.24/4.71 14:05:20
SSTOP
```

Table 4.2(d)

```
**********************************************************

         SUMMARY OF CARGO STATISTICS

MEAN CARGO TARDINESS                    =       5.90
STD.DEV. CARGO TARDINESS                =       8.03
MEAN WEIGHTED CARGO TARDINESS           =   35697
STD.DEV. WEIGHTED CARGO TARDINESS       =   66271
PERCENT CARGOES DELIVERED ON-TIME       =      50.00 %
PERCENT CARGOES DELIVERED LATE          =      45.00 %
PERCENT COMPLETELY DELIVERED CARGOES    =      85.00 %
PERCENT PARTIALLY DELIVERED CARGOES     =      10.00 %
PERCENT UNDELIVERED CARGOES             =       5.00 %
PERCENT TONS DELIVERED ON-TIME          =      47.75 %
PERCENT TONS DELIVERED LATE             =      34.72 %
PERCENT TONS UNDELIVERED                =      17.53 %

**********************************************************

**********************************************************

            SUMMARY OF SHIP STATISTICS

MEAN ROUTE CIRCUITRY                    =       0.89
MEAN WEIGHTED ROUTE CIRCUITRY           =       0.82
MEAN SHIP UTILIZATION                   =       2.87 %
MEAN DISTANCE TRAVELED                  =   21985
STD.DEV. OF DISTANCE TRAVELED           =    9001
MEAN TON-MILES                          =  1.34E+08
STD.DEV. OF TON-MILES                   =  1.04E+08
MEAN SHIPPING DISTANCE                  =    7653
MEAN PORT DELAY PER STOP                =       4.167
STD.DEV. OF PORT DELAY PER STOP         =      12.980
MEAN PORT DELAY PER SHIP                =      25.000
STD.DEV. OF PORT DELAY PER SHIP         =      23.317
MEAN PERCENT OF TIME SPENT IN PORT      =      45.34 %
MEAN ROUTE TIME                         =      55.14
STD.DEV. OF ROUTE TIME                  =      26.09
R; T=2.08/2.95 12:13:04
SSTOP
```

Table 4.3: 7 Ships, Data Set B

```
**********************************************************

          SUMMARY OF CARGO STATISTICS

MEAN CARGO TARDINESS                    =       5.45
STD.DEV. CARGO TARDINESS                =       8.10
MEAN WEIGHTED CARGO TARDINESS           =   34423
STD.DEV. WEIGHTED CARGO TARDINESS       =   80202
PERCENT CARGOES DELIVERED ON-TIME       =      60.00 %
PERCENT CARGOES DELIVERED LATE          =      40.00 %
PERCENT COMPLETELY DELIVERED CARGOES    =      90.00 %
PERCENT PARTIALLY DELIVERED CARGOES     =      10.00 %
PERCENT UNDELIVERED CARGOES             =        0.0 %
PERCENT TONS DELIVERED ON-TIME          =      49.43 %
PERCENT TONS DELIVERED LATE             =      32.10 %
PERCENT TONS UNDELIVERED                =      18.47 %

**********************************************************

**********************************************************

          SUMMARY OF SHIP STATISTICS

MEAN ROUTE CIRCUITRY                    =       1.19
MEAN WEIGHTED ROUTE CIRCUITRY           =       1.26
MEAN SHIP UTILIZATION                   =       4.68 %
MEAN DISTANCE TRAVELED                  =   27878
STD.DEV. OF DISTANCE TRAVELED           =   10007
MEAN TON-MILES                          =  1.79E+08
STD.DEV. OF TON-MILES                   =  1.11E+08
MEAN SHIPPING DISTANCE                  =   10313
MEAN PORT DELAY PER STOP                =       1.881
STD.DEV. OF PORT DELAY PER STOP         =       5.460
MEAN PORT DELAY PER SHIP                =      11.286
STD.DEV. OF PORT DELAY PER SHIP         =       8.939
MEAN PERCENT OF TIME SPENT IN PORT      =      18.46 %
MEAN ROUTE TIME                         =      61.14
STD.DEV. OF ROUTE TIME                  =      18.80
R; T=2.15/3.01 12:15:55
SSTOP
```

Table 4.4: 7 Ships, Data Set B

```
**********************************************************

          SUMMARY OF CARGO STATISTICS

MEAN CARGO TARDINESS                    =        5.35
STD.DEV. CARGO TARDINESS                =        7.95
MEAN WEIGHTED CARGO TARDINESS           =    36515
STD.DEV. WEIGHTED CARGO TARDINESS       =    75687
PERCENT CARGOES DELIVERED ON-TIME       =       55.00 %
PERCENT CARGOES DELIVERED LATE          =       45.00 %
PERCENT COMPLETELY DELIVERED CARGOES    =       90.00 %
PERCENT PARTIALLY DELIVERED CARGOES     =       10.00 %
PERCENT UNDELIVERED CARGOES             =        0.0 %
PERCENT TONS DELIVERED ON-TIME          =       48.23 %
PERCENT TONS DELIVERED LATE             =       42.82 %
PERCENT TONS UNDELIVERED                =        8.96 %

**********************************************************

**********************************************************

          SUMMARY OF SHIP STATISTICS

MEAN ROUTE CIRCUITRY                    =        1.13
MEAN WEIGHTED ROUTE CIRCUITRY           =        1.14
MEAN SHIP UTILIZATION                   =        6.80 %
MEAN DISTANCE TRAVELED                  =    28464
STD.DEV. OF DISTANCE TRAVELED           =     8105
MEAN TON-MILES                          =  1.90E+08
STD.DEV. OF TON-MILES                   =  8.80E+07
MEAN SHIPPING DISTANCE                  =     9810
MEAN PORT DELAY PER STOP                =        1.682
STD.DEV. OF PORT DELAY PER STOP         =        4.850
MEAN PORT DELAY PER SHIP                =       10.571
STD.DEV. OF PORT DELAY PER SHIP         =        7.678
MEAN PERCENT OF TIME SPENT IN PORT      =       17.66 %
MEAN ROUTE TIME                         =       59.86
STD.DEV. OF ROUTE TIME                  =       12.92
R; T=2.11/2.98 12:18:32
SSTOP
```

Table 4.5: 7 Ships, Data Set B

```
**********************************************************

          SUMMARY OF CARGO STATISTICS

MEAN CARGO TARDINESS                    =        8.55
STD.DEV. CARGO TARDINESS                =       13.22
MEAN WEIGHTED CARGO TARDINESS           =   51941
STD.DEV. WEIGHTED CARGO TARDINESS       =   97347
PERCENT CARGOES DELIVERED ON-TIME       =       50.00 %
PERCENT CARGOES DELIVERED LATE          =       40.00 %
PERCENT COMPLETELY DELIVERED CARGOES    =       75.00 %
PERCENT PARTIALLY DELIVERED CARGOES     =       15.00 %
PERCENT UNDELIVERED CARGOES             =       10.00 %
PERCENT TONS DELIVERED ON-TIME          =       33.41 %
PERCENT TONS DELIVERED LATE             =       34.44 %
PERCENT TONS UNDELIVERED                =       32.15 %

**********************************************************


**********************************************************

              SUMMARY OF SHIP STATISTICS

MEAN ROUTE CIRCUITRY                    =        1.00
MEAN WEIGHTED ROUTE CIRCUITRY           =        1.12
MEAN SHIP UTILIZATION                   =       11.47 %
MEAN DISTANCE TRAVELED                  =   29784
STD.DEV. OF DISTANCE TRAVELED           =    8447
MEAN TON-MILES                          =  2.00E+08
STD.DEV. OF TON-MILES                   =  1.15E+08
MEAN SHIPPING DISTANCE                  =    9885
MEAN PORT DELAY PER STOP                =       1.158
STD.DEV. OF PORT DELAY PER STOP         =       3.796
MEAN PORT DELAY PER SHIP                =       8.800
STD.DEV. OF PORT DELAY PER SHIP         =       7.014
MEAN PERCENT OF TIME SPENT IN PORT      =      12.43 %
MEAN ROUTE TIME                         =      70.80
STD.DEV. OF ROUTE TIME                  =      21.57
R; T=2.57/3.68 12:10:29
```

Table 4.6: 5 Ships, Data Set B

```
**********************************************************

         SUMMARY OF CARGO STATISTICS

MEAN CARGO TARDINESS                    =       11.65
STD.DEV. CARGO TARDINESS                =       16.67
MEAN WEIGHTED CARGO TARDINESS           =    73126
STD.DEV. WEIGHTED CARGO TARDINESS       =   106733
PERCENT CARGOES DELIVERED ON-TIME       =       30.00 %
PERCENT CARGOES DELIVERED LATE          =       55.00 %
PERCENT COMPLETELY DELIVERED CARGOES    =       75.00 %
PERCENT PARTIALLY DELIVERED CARGOES     =       10.00 %
PERCENT UNDELIVERED CARGOES             =       15.00 %
PERCENT TONS DELIVERED ON-TIME          =       20.76 %
PERCENT TONS DELIVERED LATE             =       57.21 %
PERCENT TONS UNDELIVERED                =       22.03 %

**********************************************************


**********************************************************

             SUMMARY OF SHIP STATISTICS

MEAN ROUTE CIRCUITRY                    =        1.11
MEAN WEIGHTED ROUTE CIRCUITRY           =        1.12
MEAN SHIP UTILIZATION                   =       10.80 %
MEAN DISTANCE TRAVELED                  =    31630
STD.DEV. OF DISTANCE TRAVELED           =    12247
MEAN TON-MILES                          =  2.15E+08
STD.DEV. OF TON-MILES                   =  9.46E+07
MEAN SHIPPING DISTANCE                  =     9235
MEAN PORT DELAY PER STOP                =        1.237
STD.DEV. OF PORT DELAY PER STOP         =        4.444
MEAN PORT DELAY PER SHIP                =        9.400
STD.DEV. OF PORT DELAY PER SHIP         =        9.317
MEAN PERCENT OF TIME SPENT IN PORT      =       12.53 %
MEAN ROUTE TIME                         =       75.00
STD.DEV. OF ROUTE TIME                  =       32.26
R; T=2.66/3.90 13:06:33
SSTOP
```

Table 4.7: 5 Ships, Data Set B

```
**********************************************************

         SUMMARY OF CARGO STATISTICS

MEAN CARGO TARDINESS                    =       3.40
STD.DEV. CARGO TARDINESS                =       5.70
MEAN WEIGHTED CARGO TARDINESS           =   14240
STD.DEV. WEIGHTED CARGO TARDINESS       =   26227
PERCENT CARGOES DELIVERED ON-TIME       =      45.00 %
PERCENT CARGOES DELIVERED LATE          =      35.00 %
PERCENT COMPLETELY DELIVERED CARGOES    =      70.00 %
PERCENT PARTIALLY DELIVERED CARGOES     =      10.00 %
PERCENT UNDELIVERED CARGOES             =      20.00 %
PERCENT TONS DELIVERED ON-TIME          =      24.15 %
PERCENT TONS DELIVERED LATE             =      33.91 %
PERCENT TONS UNDELIVERED                =      41.95 %

**********************************************************

**********************************************************

             SUMMARY OF SHIP STATISTICS

MEAN ROUTE CIRCUITRY                    =       0.99
MEAN WEIGHTED ROUTE CIRCUITRY           =       1.17
MEAN SHIP UTILIZATION                   =       9.62 %
MEAN DISTANCE TRAVELED                  =   24569
STD.DEV. OF DISTANCE TRAVELED           =   10590
MEAN TON-MILES                          =  1.53E+08
STD.DEV. OF TON-MILES                   =  4.77E+07
MEAN SHIPPING DISTANCE                  =    8815
MEAN PORT DELAY PER STOP                =       1.500
STD.DEV. OF PORT DELAY PER STOP         =       5.218
MEAN PORT DELAY PER SHIP                =      10.200
STD.DEV. OF PORT DELAY PER SHIP         =      10.663
MEAN PERCENT OF TIME SPENT IN PORT      =      17.17 %
MEAN ROUTE TIME                         =      59.40
STD.DEV. OF ROUTE TIME                  =      27.10
R; T=2.55/3.74 13:09:23
SSTOP
```

Table 4.8: 5 Ships, Data Set B

dropped by more than 13%. This indicated that the scheduling constraints were too tight for such a small number of ships to handle with reasonable ship utilization. We therefore further refined the cargo sizes, as indicated in Column C, Table 4.1, to form Data Set C. In this data set no cargoes require multiple ship-trips.

Summary statistics from this third data set appear in Tables 4.9, 4.10 and 4.11. They show a substantial improvement in performance for the 5-ship case. In the best of three seed-cargo configurations, 96% of the tons and 95% of the cargoes are delivered. For this case ship utilization is 6.2%, indicating many more cargoes could be transported with appropriate scheduling requirements. In addition, the percentage of on-time delivered cargoes hovered around the 40% level, which is relatively low. We opted not to try seven-ships of this case because of the already-low ship utilization.

A closer look at the schedules associated with the above runs reveals several interesting observations. Take for instance ship # 6 of run # 1 (displayed in Table 4.2). This ship picks up cargo # 6 from Charleston, S.C. on day 25 and delivers that cargo in Malaysia on day 47. It subsequently travels on ballast to Okinawa where on day 57 it loads three cargoes: Cargo # 17, and parts of cargoes # 15 and # 16, both of which are large-size and have to be split. Cargo # 15 is split in two, 10,850 tons carried by the above ship (#6), and 13,800 tons carried by ship # 5 several days later (see also Table 4.2). Cargo # 16 is carried entirely by ship # 6, but in four (4) rountrips (Okinawa to Malaysia), because of its even larger size (51,880 tons, compared to a capacity of 19,125 tons for ship #6). Ship # 6 completes its trip on day 88, after

```
**********************************************************

         SUMMARY OF CARGO STATISTICS

MEAN CARGO TARDINESS                    =       8.45
STD.DEV. CARGO TARDINESS                =       8.77
MEAN WEIGHTED CARGO TARDINESS           =   53377
STD.DEV. WEIGHTED CARGO TARDINESS       =   90051
PERCENT CARGOES DELIVERED ON-TIME       =      30.00 %
PERCENT CARGOES DELIVERED LATE          =      70.00 %
PERCENT COMPLETELY DELIVERED CARGOES    =      95.00 %
PERCENT PARTIALLY DELIVERED CARGOES     =       5.00 %
PERCENT UNDELIVERED CARGOES             =        0.0 %
PERCENT TONS DELIVERED ON-TIME          =      21.12 %
PERCENT TONS DELIVERED LATE             =      69.30 %
PERCENT TONS UNDELIVERED                =       9.59 %

**********************************************************

**********************************************************

             SUMMARY OF SHIP STATISTICS

MEAN ROUTE CIRCUITRY                    =       0.98
MEAN WEIGHTED ROUTE CIRCUITRY           =       0.87
MEAN SHIP UTILIZATION                   =       1.80 %
MEAN DISTANCE TRAVELED                  =   30616
STD.DEV. OF DISTANCE TRAVELLED          =    9769
MEAN TON-MILES                          =  1.62E+08
STD.DEV. OF TON-MILES                   =  8.29E+07
MEAN SHIPPING DISTANCE                  =    7513
MEAN PORT DELAY PER STOP                =       0.667
STD.DEV. OF PORT DELAY PER STOP         =       4.165
MEAN PORT DELAY PER SHIP                =       5.600
STD.DEV. OF PORT DELAY PER SHIP         =      11.971
MEAN PERCENT OF TIME SPENT IN PORT      =       7.53 %
MEAN ROUTE TIME                         =      74.40
STD.DEV. OF ROUTE TIME                  =      24.32
R; T=2.45/3.53 13:19:05
$STOP
```

Table 4.9: 5 Ships, Data Set C

```
**********************************************************

         SUMMARY OF CARGO STATISTICS

MEAN CARGO TARDINESS                    =       11.00
STD.DEV. CARGO TARDINESS                =       19.34
MEAN WEIGHTED CARGO TARDINESS           =    83080
STD.DEV. WEIGHTED CARGO TARDINESS       =   175713
PERCENT CARGOES DELIVERED ON-TIME       =       45.00 %
PERCENT CARGOES DELIVERED LATE          =       50.00 %
PERCENT COMPLETELY DELIVERED CARGOES    =       95.00 %
PERCENT PARTIALLY DELIVERED CARGOES     =        0.0 %
PERCENT UNDELIVERED CARGOES             =        5.00 %
PERCENT TONS DELIVERED ON-TIME          =       30.21 %
PERCENT TONS DELIVERED LATE             =       65.74 %
PERCENT TONS UNDELIVERED                =        4.05 %

**********************************************************


**********************************************************

              SUMMARY OF SHIP STATISTICS

MEAN ROUTE CIRCUITRY                    =        1.15
MEAN WEIGHTED ROUTE CIRCUITRY           =        1.06
MEAN SHIP UTILIZATION                   =        6.18 %
MEAN DISTANCE TRAVELED                  =    35109
STD.DEV. OF DISTANCE TRAVELED           =    15253
MEAN TON-MILES                          =  2.20E+08
STD.DEV. OF TON-MILES                   =  1.20E+08
MEAN SHIPPING DISTANCE                  =     9607
MEAN PORT DELAY PER STOP                =        1.100
STD.DEV. OF PORT DELAY PER STOP         =        3.529
MEAN PORT DELAY PER SHIP                =        8.800
STD.DEV. OF PORT DELAY PER SHIP         =        6.058
MEAN PERCENT OF TIME SPENT IN PORT      =       10.81 %
MEAN ROUTE TIME                         =       81.40
STD.DEV. OF ROUTE TIME                  =       36.77
R; T=2.34/3.38 13:22:33
SSTOP
```

Table 4.10: 5 Ships, Data Set C

```
**********************************************************

         SUMMARY OF CARGO STATISTICS

MEAN CARGO TARDINESS                    =       9.00
STD.DEV. CARGO TARDINESS                =      14.76
MEAN WEIGHTED CARGO TARDINESS           =   48146
STD.DEV. WEIGHTED CARGO TARDINESS       =   90576
PERCENT CARGOES DELIVERED ON-TIME       =      45.00 %
PERCENT CARGOES DELIVERED LATE          =      45.00 %
PERCENT COMPLETELY DELIVERED CARGOES    =      85.00 %
PERCENT PARTIALLY DELIVERED CARGOES     =       5.00 %
PERCENT UNDELIVERED CARGOES             =      10.00 %
PERCENT TONS DELIVERED ON-TIME          =      34.95 %
PERCENT TONS DELIVERED LATE             =      48.89 %
PERCENT TONS UNDELIVERED                =      16.16 %

**********************************************************

**********************************************************

              SUMMARY OF SHIP STATISTICS

MEAN ROUTE CIRCUITRY                    =       0.99
MEAN WEIGHTED ROUTE CIRCUITRY           =       1.12
MEAN SHIP UTILIZATION                   =      12.54 %
MEAN DISTANCE TRAVELED                  =   29644
STD.DEV. OF DISTANCE TRAVELED           =    9223
MEAN TON-MILES                          =  1.96E+08
STD.DEV. OF TON-MILES                   =  9.84E+07
MEAN SHIPPING DISTANCE                  =    9807
MEAN PORT DELAY PER STOP                =       2.105
STD.DEV. OF PORT DELAY PER STOP         =       8.036
MEAN PORT DELAY PER SHIP                =      16.000
STD.DEV. OF PORT DELAY PER SHIP         =      17.875
MEAN PERCENT OF TIME SPENT IN PORT      =      22.10 %
MEAN ROUTE TIME                         =      72.40
STD.DEV. OF ROUTE TIME                  =      26.43
R; T=2.46/3.55 13:24:19
SSTOP
```

Table 4.11: 5 Ships, Data Set C

making these four roundtrips from Okinawa to Malaysia. Notice that this ship travels from the East Coast of the U.S. to the Pacific practically empty, since cargo #6 is only 30 tons. This is due to the fact that cargo # 6 was assigned arbitrarily to ship # 6 as a seed assignment by the user, and that no other cargoes were assigned to travel on the same ship on its way to the Pacific because of incompatible schedules.

Similar interesting schedule patterns were observed in other runs of MORSS.

In the rest of this section we further analyze the performance of MORSS based on the results of the previous discussion. First, it is evident that the performance of the system depends highly on the particular data set being used. Much of the discussion in the first part of this section focused on tailoring the data set to a small fleet size. It is unclear what the a priori limits for feasibility are. For example, cargoes #2 and #9 seem to be "difficult" cargoes in the sense that they are often at least partially unscheduled. Many factors are involved. Boundary conditions, cargo size, seed selection and interactions between large cargoes may account for much of the difficulty. In addition, the parameters involved in the utility functions need detailed calibration themselves (see also Chapter 5).

To aid in the analysis of the runs, MORSS computed a large variety of statistics at the end of each run. The most important ones are summarized in Table 4.12, for each of the runs. These statistics are the measures by which we evaluated the performance of the system. The most important of the cargo statistics are the percentages of cargoes (delivered) on time and tons (delivered) on time. These most clearly

RUN NO.	DATA SET	# SHIPS	SEED CARGOES FOR SHIPS 1,2,3,..	% CARGOES ON TIME	% CARGOES DELIVERED	% TONS ON TIME	% TONS DELIVERED	MEAN TARDINESS FOR LATE CARGOES (days)	SHIP UTILIZATION %
1	A	7	1,2,3,4,5,6,7	35.00	75.00	10.84	27.10	8.80	11.93
2	B	7	1,3,5,7,9,11,13	50.00	85.00	47.75	82.47	5.90	2.87
3	B	7	1,3,5,7,9,2,6	60.00	90.00	49.43	82.53	5.45	4.68*
4	B	7	1,3,5,8,9,2,6	55.00	90.00	48.23	91.05	5.35	6.80
5	B	5	1,3,5,7,9	50.00	75.00	33.41	67.85	8.55	11.47
6	B	5	1,2,5,8,9	30.00	75.00	20.76	77.97	11.65	10.80
7	B	5	2,4,6,7,8	45.00	70.00	24.15	58.06	3.46	9.62
8	C	5	1,2,3,4,5	30.00	95.00	21.12	90.42	8.45	1.80*
9	C	5	1,3,5,6,8	45.00	95.00	30.21	95.95	11.00	6.18*
10	C	5	8,7,5,3,1	45.00	85.00	34.95	83.84	9.00	12.54

TABLE 4.12: Summary of Results

reflect the primary concern of the MSC to deliver cargoes on time.

Among the ship-related statistics, route circuity, ship utilization and percent of time spent in port are the most important. By "route circuity" we mean the ratio of the total distance traveled by a ship divided by the sum of POE-POD distances for all cargoes carried by the ship. Thus, a low circuity (less than half) means the ship carries many cargoes most of the time, while a high circuity (more than 1.5) means the ship carries few cargoes at a time, and is deadheading a good part of the time also. These measures are important as indications of the efficiency at which the system operates.

We now turn to a discussion of the correlation between the statistics generated by MORSS. Plots of some of these appear in Figures 4.3 to 4.6. From Figures 4.5 and 4.6 we see that ship utilization and the percentage of tons delivered are not correlated with the percentage of tons delivered on time. Figure 4.3 shows a strong linear correlation between the percentage of on-time cargoes and tons. Figure 4.4 indicates the percentage of cargoes on time is close to being independent of the percentage delivered. These preliminary results indicate that ship utilization is not a good performance measure for problems with tight or infeasible scheduling requirements. This result will be checked more closely with a larger feasible set of cargo movement requirements during the calibration procedure.

We propose several bounds on system performance. First, ship utilization has a maximum of the order of 50% with the given POE-POD structure. This is because of traveling on ballast from POD to POE. Practically speaking, this bound is not achievable either, because of the

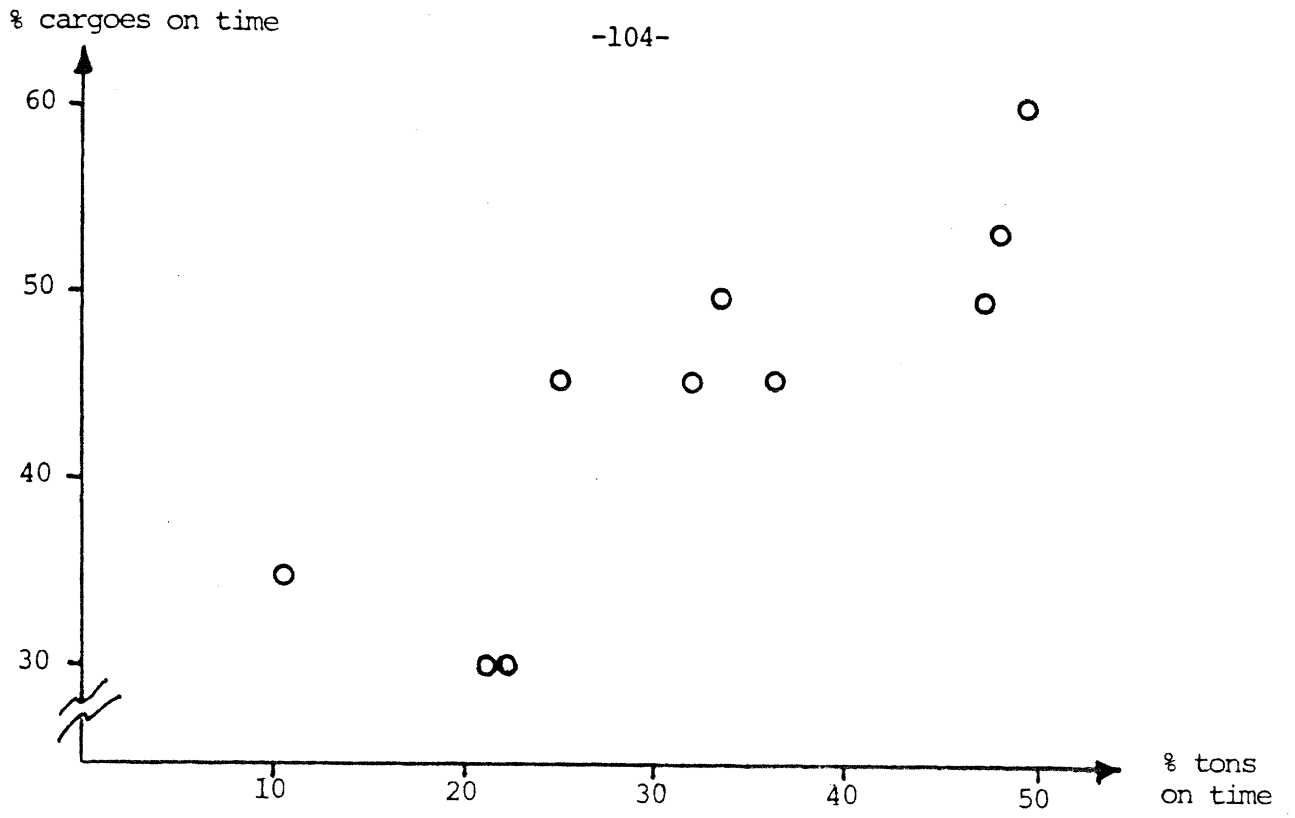


Figure 4.3

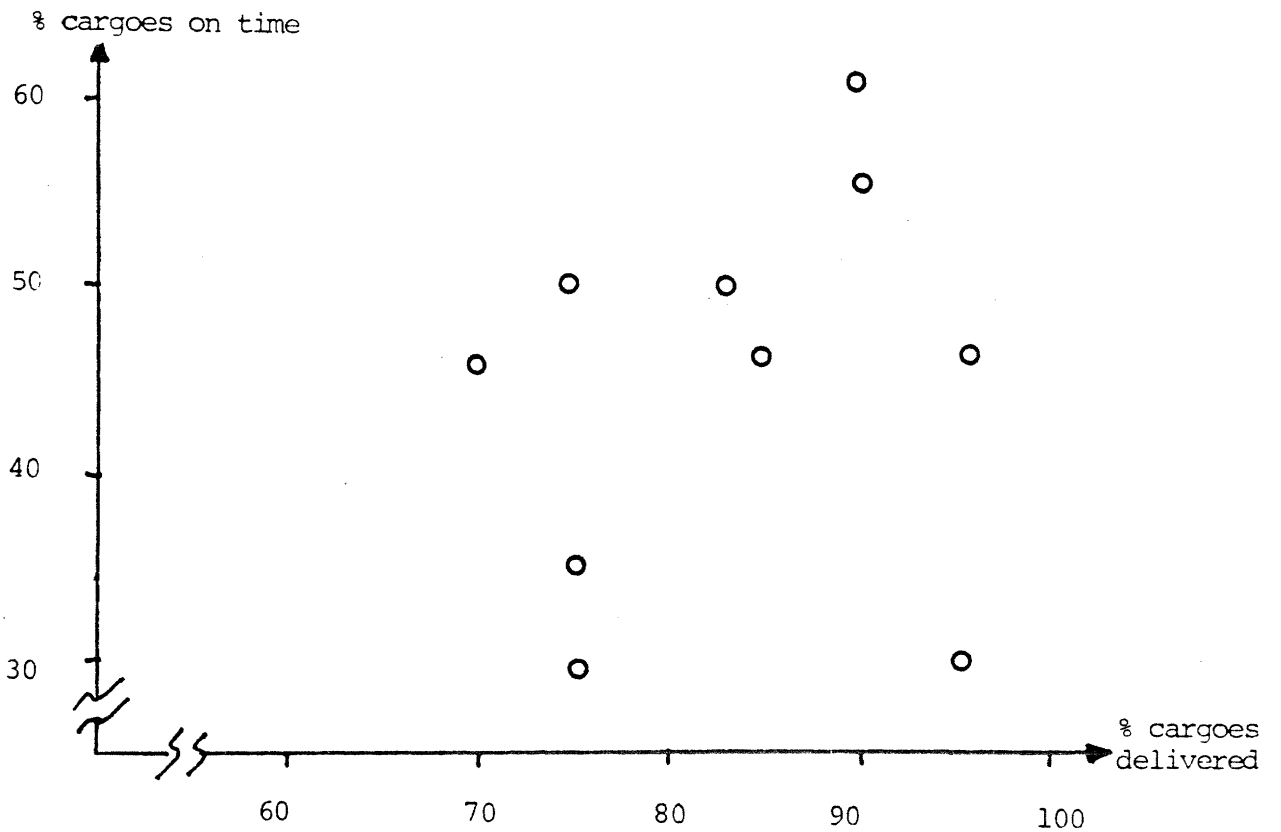


Figure 4.4

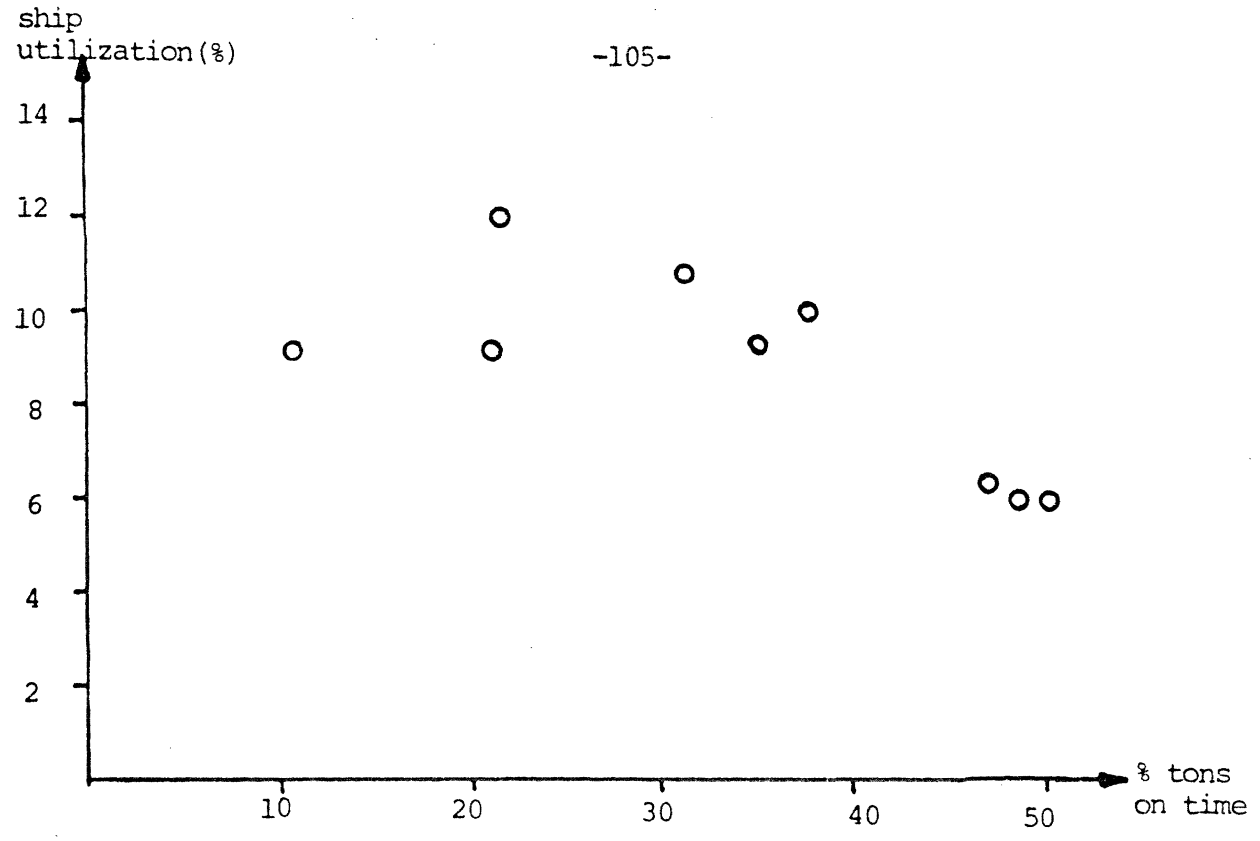


Figure 4.5

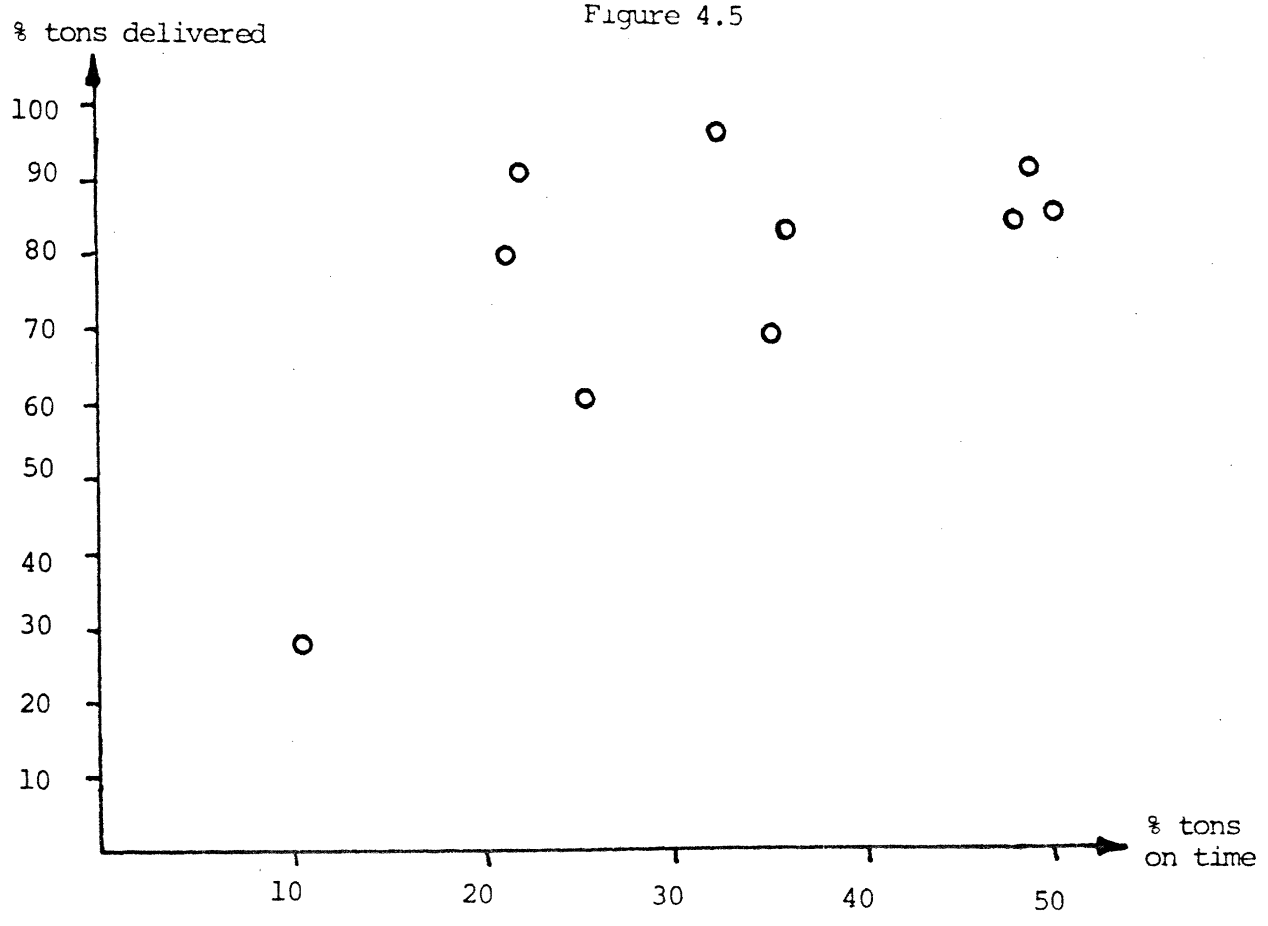


Figure 4.6

necessity of multiple cargoes on the same ship. In this case the ship is not full between deliveries. Second, route circuity has a maximum of 2.0, corresponding to the case where a ship carries only one cargo at a time. Both of these quantities are bounded below by zero. Third, each data-set has its own particular structure. In many cases this results in bounds on the number of cargoes or tons that can be delivered on time.

Given these bounds, it is worthwhile to note that the best ship utilization achieved in these runs had a value of 12.54% (in run # 10). This means that the ships in that sealift problem instance averaged about 25% full while transporting cargo. This is a reasonable performance for such a tight problem, where the average late cargo was 9 days overdue. This points out the difficulty in establishing norms for the expected performance of the system under MORSS or any scheduling algorithm. This issue will be further explored in a continuing phase of the project (see also Chapter 5).

CHAPTER 5

CONCLUDING REMARKS

5.1 Introduction

This chapter provides some additional details on miscellaneous other activities related to this project (section 5.2) and concludes by describing directions for further research on the MSC operational problem (section 5.3).

5.2 Other Project-Related Written Products

There have been several additional self-contained products, written by members of the MIT project team, and related to this project. These include an annotated bibliography, - by Thompson (1983) -, two M.Sc. theses, - by Badjis (1982) and Jeng (1984), and a PhD thesis, - by Kim (1985).

In Thompson (1983), a bibliography of about 70 references in this general problem area is described. This bibliography is maintained in the MIT project library and updated at regular time intervals.

The M.Sc thesis of Bardjis (1982) was completed a few months before this project was initiated, and hence cannot be considered an "official" product of the project. However, we include it here because to our knowledge it was the first attempt to look into the MSC operational problem from a rigorous analytical standpoint. The thesis developed exact mathematical programming formulations for a spectrum of variants of the MSC problem, by examining a variety of objective functions, constraints, etc. Upon initiation of the project, the MIT team decided not to use an exact

approach (for the reasons outlined in Chapter 2), and hence these formulations were no longer pursued.

The M.Sc. thesis of Jeng (1984) took the "sequential insertion" algorithm developed by Jaw et al (1984) for the multi-vehicle many-to-many advance-request dial-a-ride problem with time windows, and attempted to adapt it to the MSC operational problem. As described in Chapter 2, the "sequential insertion" algorithm is a very efficient procedure for scheduling a fleet of vehicles in a dial-a-ride environment, and computational experience with the procedure has been very satisfactory. Jeng's work dealt with necessary modifications in the procedure so that it could be applied to a version of the MSC operational problem. A generic computer-program was written to that effect, but was never implemented on the MSC-supplied or other data.

The PhD thesis of Kim (1985) shed light on some very interesting theoretical aspects of the MSC problem. Specifically, it recognized that in the absence of time constraints (EPT's, EDT's and LDT's), the routing subproblem of the MSC problem becomes usually very easy to solve, given that ports are located "along the shoreline". Kim them defined the "shoreline" distance metric as a special case of the Euclidean distance metric and imposed time constraints on the problem. He then developed a class of heuristic (approximate) algorithms for the single-ship "shoreline" problem with time constraints, and derived the worst-case performance of these algorithms (in terms of deviation from the theoretical optimum). For the special case in which all ports are located on a straight line, he showed that depending on the objective function and the constraints of the problem, the problem can be either solved exactly by a "polynomial"

algorithm, or belongs to the class of "NP-complete" problems (for which no polynomial algorithm is known to exist.) Kim's work has opened some very interesting directions for further research on this class of prolems.

5.3 Conclusions and Directions for Further Research

We conclude the main body of the report with our conclusions and our ideas on directions for further research in this area.

To our knowledge, today the MORSS algorithm represents the only methodology that has been designed and developed specifically for the MSC operational routing and scheduling problem. As stated in the project's statement of objectives (see Chapter 1), the MIT team set forth to investigate a class of sealift routing and scheduling problems, develop, analyze and test solution algorithms for such problems, and work with the MSC and others so as to increase the level of knowledge of these problems. As a result of our efforts within the past two and a half years, we feel that progress in the project has been in accordance with the above objectives and with the schedule we had proposed to accomplish them. Not only do we know much more about the structure and complexity of this problem now than we did when this project was starting, but we do possess a methodology and an associated computer program that can form the basis for future implementation by the MSC. We also feel that we have made progress regarding the stated longer-term objectives of this project, that is, to develop a procedure that could be ultimately implemented by the MSC, enhance the state of the art in the solution of complex, large-scale transportation problems, and advance the state of knowledge in interactive user-friendly algorithms. In fact, we feel that many of the principles and

methods we have adopted for this project could be used in other environments as well.

There is no question that our investigation to date has left considerable room for both further analysis of the MSC operational problem and algorithmic development of MORSS. In the following we describe a set of possible follow-on activities which we consider essential for further progress in this area. These activities would build upon the work accomplished to date, by targeting specific issues that have been either addressed only superficially, or not addressed at all by this project thus far. We should emphasize that all of these activities intentionally fall into the category of research, creation of new knowledge, and methodology development, as opposed to real-world implementation of a software package (a possible exception is Task 6, which addresses implementation issues to some extent.)

The description of these activities goes as follows:

Task 1: Investigation and Caliberation of Alternative Utility Functions:

Although considerable effort has been already spent to come up with utility functional forms that make sense for this problem, it is conceivable that other classes of functions could be used so as to (a) model the MSC decision process more realistically, (b) decrease the required computational effort and (c) achieve improved solutions. It is clear that such a task would involve an increased degree of interaction with MSC personnel, so as to incorporate their suggestions and other feedback into the algorithm. By "calibration" we mean the determination of the most appropriate values of the parameters of those utility functions. Since those values are user-inputs, interaction with the MSC is essential

here too.

Task 2: Investigation of Cargo Assignment Interactions: MORSS currently assigns utilities to potential cargo-ship assignment pairs without taking into account possible interactions of unassigned cargoes with one another. Thus, it is conceivable that while each of cargoes i and j is by itself a good assignment for ship k, assigning both of them to the same ship is not. MORSS currently handles this problem by (a) limiting the number of tentative cargo assignments simultaneously made to a ship at each iteration, and (b) by "deassigning" those cargoes from a ship's list of tentative assignments that do not interact well. This task would investigate other, computationally more efficient methods for taking such possible interactions into account. Possible approaches in this context include the development of a "pre-opt" procedure which would eliminate unlikely arcs in the transportation network, and/or the construction of a "swapper, post-opt" heuristic which would eliminate poor combinations of cargoes once the assignments are made.

Task 3: Development of More Sophisticated Seed Selection Methods: Seed selection is considered to be an important step in MORSS. The current procedure accomplishes such selection by solving a one-to-one assignment problem involving ship and cargo pairs whose utilities have been computed. We would like to explore and test alternative seed selection methods. Those might include selecting seeds so as to maximize some measure of "spread-out-edness", a measure which, by definition, involves interaction among seed cargoes. A subtask here might consist of developing more advanced aggregation/clustering methods to facilitate the seed selection process. Finally, we would like to investigate the sensitivity ofthe

overall schedule to seed selection rules.

Task 4: Modeling of Queueing Effects at Ports: Ship queueing at ports during periods of congestion is likely to be a very significant issue during the actual execution of the schedule. MORSS currently includes a term in the utility function that "measures" port congestion, but it is clear that this problem merits a far more extensive investigation. We propose to develop methods that predict future queueing effects at ports more accurately. We anticipate that in doing so we shall need more accurate information regarding the structure of ports and their throughput as a function of demand rate. This analysis will move away from the essentially deterministic approach that has been used so far, to the modeling of probabilistic effects. "Optimal" loading/unloading/queueing disciplines would be investigated so as to improve port utilization and reduce overall delays.

Task 5: Sensitivity Analysis: Other than user-specified parameters calibration, we would like to perform sensitivity analysis on many important categories of the input data, and answer "what if" questions that would help obtain further insights into this problem. Such data include due dates for cargoes (i.e., "what if the due date for this cargo is moved backward by one day?", "what if the NATO-controlled fleet is made available?", etc.

Task 6: Further Testing of Algorithm and User Friendly Implementation: We would like to further develop the interactive portion of MORSS. This includes the capability of modifying schedule assignments, adding/removing ships and cargoes, and a graphics feature. Our plan would be to transfer MORSS from the MIT IBM system to the new Apollo color-graphics

microcomputer at the Operations Research Center (acquired by a grant from ONR). This task would also call for extensive interaction with MSC so that the ability of the MSC to eventually use the algorithm is maximized.

At the time of the writing of this report, the MIT team is engaged in Tasks 1 and 6. All other tasks are to be left for a future phase of this project.

REFERENCES

Bardjis, C.S, 1982, "Mathematical Programming Formulations of a Sealift Scheduling Problem," M.Sc. Thesis, M.I.T., Department of Ocean Engineering, May.

Bodin, L. and T. Sexton, 1982. "The Multi-Vehicle Subscriber Dial-A-Ride Problem." Management Science Statistics Working Paper No. 82-005, University of Maryland at College Park.

Fisher, M.L., A.J. Greenfield, R. Jaikumar, P. Kedia, 1982. "Real-Time Scheduling of a Bulk Delivery Fleet: Practical Application of Lagrangian Relaxation", Report 82-10-11, Decision Sciences Dept., University of Pennsylvania.

Fisher, M.L., M. Rosenwein, 1984. "Sceduling a Fleet of Bulk Transport Ships", TIMS/ORSA meeting, San Francisco. May.

Jarvis, J.J. and H.D. Ratliff, 1980. "Interactive Heuristics for Large Scale Transportation Models", D.O.T. Contract TSC-1618, School of Industrial and Systems Engineering, Georgia Institute of Technology, August.

Jarvis, J.J. and H.D. Ratliff, 1982. "Preliminary Analysis System for Closure Optimization Planning and Evaluation (Scope),": Report No. PDRC 82-22 (for Joint Deployment Agency), School of Industrial Systems Engineering, Georgia Institute of Technology.

Jaw, J.J., A.R. Odoni, H.N. Psaraftis, N.H.M. Wilson, 1982. "A Heuristic Algorithm for the Multi-Vehicle-Many-to-Many Advance-Request Dial-A-Ride Problem, "Working Paper No. MIT-UMTA-82-3, M.I.T. June.

Jaw, J.J., A. R. Odoni, H.N. Psaraftis, N.H.M. Wilson, 1984. "A Heuristic Algorithm for the Multi-Vehcile Advance-Request Dial-A-Ride Problem With Time Windows", forthcoming in Transportation Research(B).

Jaw, J.J., 1984. "Solving Large-Scale Dial-A-Ride Vehicle Routing and Scheduling Problems", Ph.D. Thesis, Dept. of Aeronautics and Astronautics M.I.T., FTL REPORT R84-3, June.

Jeng, K.Y., 1984. "Preliminary Considerations on using an Insertion Algorithm For the Solution of a Sealift Routing and Scheduling Problem", S.M. Thesis, Dept. of Ocean Engineering, M.I.T., May.

Kaskin, J. 1981. "Suggestions for the Improvement of the MSC Strategic Sealift Contingency Planning System (SEACOP), Point Paper, MSC.

Kim, T.U., 1985. "Solution Algorithms for Sealift Routing Scheduling Problems", Ph.D. Thesis, Dept. of Ocean Engineering, M.I.T., May.

Kim, T.U., H.N. Psaraftis, M.M. Solomon, 1985. "Time Constrained Ship

Routing and Scheduling Problems with an Easy Routing Structure", EURO-VII Congress, Bologna, Italy, June.

Orlin, J.B., 1983a. "On the Simplex Algorithm for Networks and Generalized Networks", Working Paper No. 1467-83, Sloan School of Management, M.I.T.

Orlin, J.B. 1983b. "A Polynomial-Time Parametric Simplex Algorithm for the Minimum Cost Network Flow Problem", Working Paper No. 484-83. Sloan School of Management, M.I.T., September.

Orlin, J.B. and H.N. Psaraftis, 1983a. "MIT Sealist Routing and Scheduling Project: Progress Report", M.I.T., July.

Orlin, J.B. and H.N. Psaraftis, 1983b. "Solution Methods for Large-Scale Sealift Scheduling Problems", ORSA/TIMS meeting, Orlando, November.

Orlin, J.B. and H.N. Psaraftis, 1984. "An Algorithm for Emergency Sealift Routing and Scheduling Problems", TIMS/ORSA meeting, San Francisco, May.

Psaraftis, H.N., 1980. "A Dynamic Programming Solution to the Single Vehicle Many-to-Many Immediate Request Dial-A-Ride Problem", Transportation Science. Vol. 14, No. 2, May.

Psaraftis, H.N., 1983a. "An Exact Algorithm for the Single-Vehicle Many-to-Many Dial-A-Ride Problem with Time Windows", Transportation Science, Vol. 17, No. 3. August.

Psaraftis, H.N. 1983b. "Analysis of an $O(N^2)$ Heuristic for the Single Vehicle Many-to-Many Euclidean Dial-A-Ride Problem", Transportation Research, Vol. 17B, No. 2.

Psaraftis, H.N. 1983c. "k-Interchange Procedures for Local Search in a Precedence-Constrained Routing Problem", European Journal of Operations Research, Vol. 13, No. 4.

Psaraftis, H.N. and J.B. Orlin, 1982. "Analysis and solution Algorithms of Sealift Routing and Scheduling Problems", proposal to ONR, October.

Psaraftis, H.N., J.B. Orlin, D. Bienstock and P. Thompson, 1985. "The Operational Routing and Scheduling Problem of the Military Sealift Command", TIMS/ORSA Meeting, Boston. May.

SAI, 1982. "Seastrat Scheduling Algorithm for Improving Lift (SAIL) Modeling Approach Definition", report to MSC by Science Applications, Inc. September.

SEASTRAT, 1981. "SEASTRAT Development Plan", MSC. July.

Scott, J., 1982. "Department of Defense Ocean Transportation Routing and Scheduling Requirements", Cargo Ship Routing and Scheduling Symposium,

Washington, February.

Thompson, P.M., 1983. "MIT Sealift Routing and Scheduling Project: An Annotated Bibliography of References". MIT, September.

Wilson, N.H.M., H. Weissberg, 1976. "Advanced Dial-A-Ride Algorithms Research Project: Final Report", Report R70-20, MIT Dept. of Civil Engineering.

Wilson, N.H.M., N.H. Salvin, 1977. "Computer Control of the Rochester Dial-A-Ride System", Report R 77-31, MIT Dept. of Civil Engineering.

APPENDIX A

DATA STRUCTURES

A.1. Introduction

This appendix is devoted to describing the data structures used in the MORSS algorithm. The MIT team spent a considerable amount of effort to come up with a data structure design that allows for fast data manipulation, robust data cross-reference, maximum accessibility for the interactive user, and ease in future program expansion. We found the use of PASCAL very useful in that respect (computer languages such as FORTRAN may be deficient in implementing this type of data structure design).

The data structures utilized fall into 4 major categories: Complex Based, Port Based, Cargo Based, and Ship Based. These structures are (roughly) interconnected as shown in Figure A.1. These interconnections are, in fact, two-way. The next few pages elaborate on the data structures and their interconnections. For a summary, refer to charts in Figures A.2 and A.3. The following constants (assumed part of input, contained in a file) are used throughout:

NUMCMP = number of complexes

MAXCRG = maximum number of cargoes

MAXSHP = maximum number of ships

INSUP = maximum number of complexes in a super-complex

CNGPER = maximum number of "congestion periods".

DATA STRUCTURE CONNECTIONS

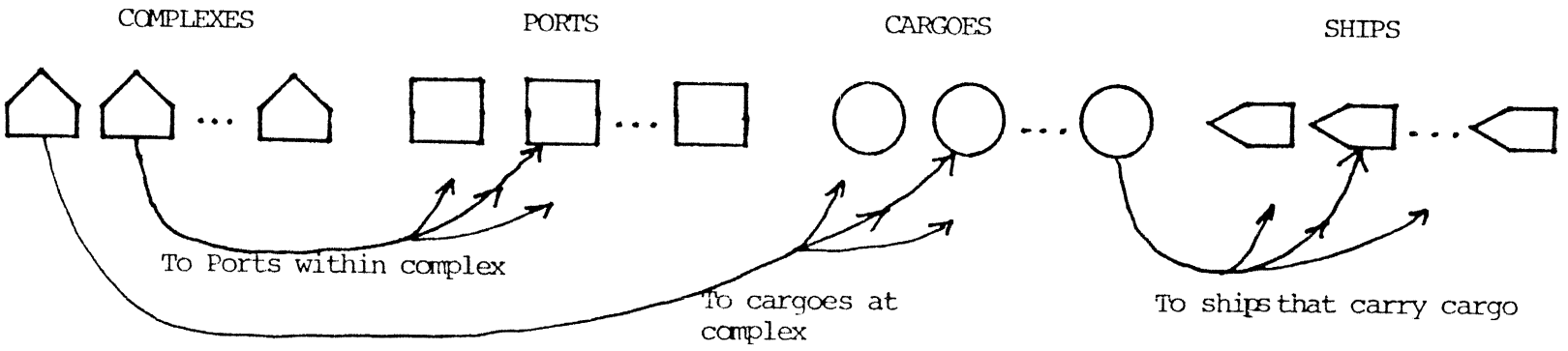


Figure A.1

A.2 Ports

The data for each port is stored in a record type UNIT. Each Unit contains the following data:

UNIT = RECORD

NAME: four characters; name of port

DEPTH: integer

LENGTH: integer

BEAM: integer

NBRTH: integer, number of berths

KIND: character, A,C,G,etc.

The ports can also be organized into arrays of up to 7 units. This is so because all complexes contain at most 7 ports. The type of array is GROUP.

GROUP = ARRAY (1..7) of UNIT

A.3 Ship Stops

For each ship we keep a linked list of that ship's stops, as they occur in time. A stop is defined as one pickup or delivery stop at a given port. If several cargoes are picked up or delivered successively, a separate stop is constructed for each. The data for each stop is contained in a record of type EVENT, pointed to by a pointer of type LINK3:

LINK3 = ↑EVENT

EVENT = RECORD

P-D: character, P or D (pickup or delivery)

CARGO: integer, code of cargo

QUANT: integer, quantity of cargo picked up/delivered

CMPL: integer, code of complex (refer to Section A.4)

PORT: integer, code of port with complex (refer to Section A.4)

DATEIN: integer, date ship finally docks at PORT

DATEOUT: integer, date ship finally leaves PORT

WAITIN: integer, wait before docking due to congestion

WAITOUT: integer, wait before leaving due to congestion

SLACK: integer, wait due to arrival previous to EPT or EDT

RMRTON: integer, masurement tons of residual capacity after stop

RSHTON: integer, short tons of residual capacity after stop

RSQFT: integer, square feet of residual capacity after stop

NEXT: link3, to next stop in schedule

PREV: link3, to previous stop in schedule

MATCH: link3, to corresponding pickup or delivery of same cargo

In addition, each stop is also addressable from the corresponding cargo record. For each cargo we keep a local linked list of all the pickups/deliveries of that cargo. Each record in the list is of type PCKUP and is pointed to by pointers of type LINK4.

LINK4 = PCKUP

PCKUP = RECORD

SHIPNO: integer, number of ship

EVNT: Link3, to pickup in SHIPNO's schedule

NXT: to next record

A.4 Cargoes

For each cargo we keep a record of type MVT pointed to by pointers of type LINK1.

LINK1 = ↑ MVT

MVT = RECORD

KEY: integer, code assigned to cargo

ORGCMP: integer, complex of origin of cargo

DESCMP: integer, destination complex of cargo

ORGPRT: integer, origin port of cargo

DESPRT: integer, destination port of cargo

EPT: integer

EDT: integer

LDT: integer

MSRTN: integer, measurement tons of cargo

SHRTN: integer, short tons of cargo

HVYLFT: integer, heavy lift code

LOADED: boolean, TRUE if we have completely loaded cargo

REMMSR: integer, measument tons still to be loaded

REMSHR: integer, short tons still to be loaded

REMSQ: integer, square feet still to be loaded

PFTYPE: integer, preferred ship type

ATTMPT: integer, number of loading attempts for cargo

FSTLDNG: link4, points to lst loading in local list

LSTLDNG: link4, points to last loading in local list

The overall set of cargoes is kept in an arrray of type MVTPTR:

MVTPTR = ARRAY (1..MAXCRG) OF LIKN1

Finally, at each complex we keep a linked list of the cargoes leaving from the complex. The records are of type CRG, pointed to by pointers of type LINK2.

LINK2 = ↑CRG

CRG = RECORD

CORR: integer, code of cargo (=Key in MVT record)

FOLL: Link2, next cargo in list

A.5 Complexes

The data for each complex is kept in a record, of type SEVRL.

SEVRL = RECORD

CODE: 5 characters, name of complex (i.e. 5EATB, etc.)

NPORTS: integer, number of ports

PORTS: GROUP, (ports at complex, Section A.2)

NCRG: integer, number of cargoes leaving from complex

FCRG: LINK2, first cargo is linked list (Section A.3)

LCRG: LINK2, last cargo in linked list

CNGST: Array (1..CNGPER) of integer, congestion level at complex

DIST: Array (1..NUMCMP) of integers, distances to all other complexes

The SEVRL records are gathered in an array of type HUBS, and the supercomplexes are stored in records of type AGGCOMP; themselves arranged in an array of type ALLAGGC.

HUBS = ARRAY (1..NUMCMP) of SERVL

AGGCOMP = RECORD

N-COMP: integer, number of complexes

INDEX: Array (1...INSUP) of integer (indices of the complexes as they appear in HUBS)

ALLAGGC: ARRAY (1..NUMCMP) of AGGCOMP

A.6 Ships

Each ship's data is kept in a record of type BOAT, pointed to by a pointer of type LINK5

LINK5 = ↑BOAT

BOAT = RECORD

CODE: integer, code assigned to ship

SHPT: integer, ship type

DRAFT: integer

LNGTH: integer

BEEM: integer, beam (deliberate mispelling)

BOOM: integer

SPEED: integer

MTNCAP: integer, metric ton capacity

STNCAP: integer, short ton capacity

FTCAP: integer, square feet capacity

FSTOP: LINK3, first stop in schedule

LSTOP: LINK3, last stop in schedule

NXTAV: ARRAY (1..2) of integer, current fully loaded period

AVO: ARRAY (1..NUMCMP) of integer, time to complex

ID: integer

GRID: integer

NISC: integer

FLEET: integer

YEAR: integer

CRGUTIL: integer, utility of last assignment

The pointers to the ship records are kept in an array

BOATPT = ARRAY (1..MAXSHO) OF LINK5

A.7 An Illustration

A graphical representation of cargoes and ships data structures appears in Figure A.2. In this figure, cargo j leaves from complex O_j and goes to complex D_j. This cargo is split (totally) between ships I and H.

Similarly, cargo k goes from O_k to D_k and is being carried - only partially - by ship I.

Ship I's only stops are to pick up and deliver parts of cargoes j and k. A graphical representation of the data structures for complexes, ports and cargoes appears in Figure A.3. In that figure, cargoes j_1, j_2, j_3 and j_4 all leave from complex I and from ports 1,3,2,2, respectively at complex I.

A.8 Some additional data structures and common names of variables

During a run of the optimization algorithm for network flows (see Chapter 3 and Appendix B for further details), a ship may be assigned at most 4 cargoes. (4 could be changed to any desired constant). We keep track of the assignments, for a given ship, in records of type PRSHP.

PRSHP = RECORD

SNUM: integer, number of ships

CNUMS: ARRAY (1..4) of integers, cargoes assigned to ship

LOSTS: ARRAY (1..4) of integers, costs of assignment

PICKS: ARRAY (1..4) of LINK3, pickups of assignment

DELVS: ARRAY (1..4) of LINK3, deliveries of assignment

COUNT: integer, number of cargoes actually assigned to ship

Constants: NU = 50

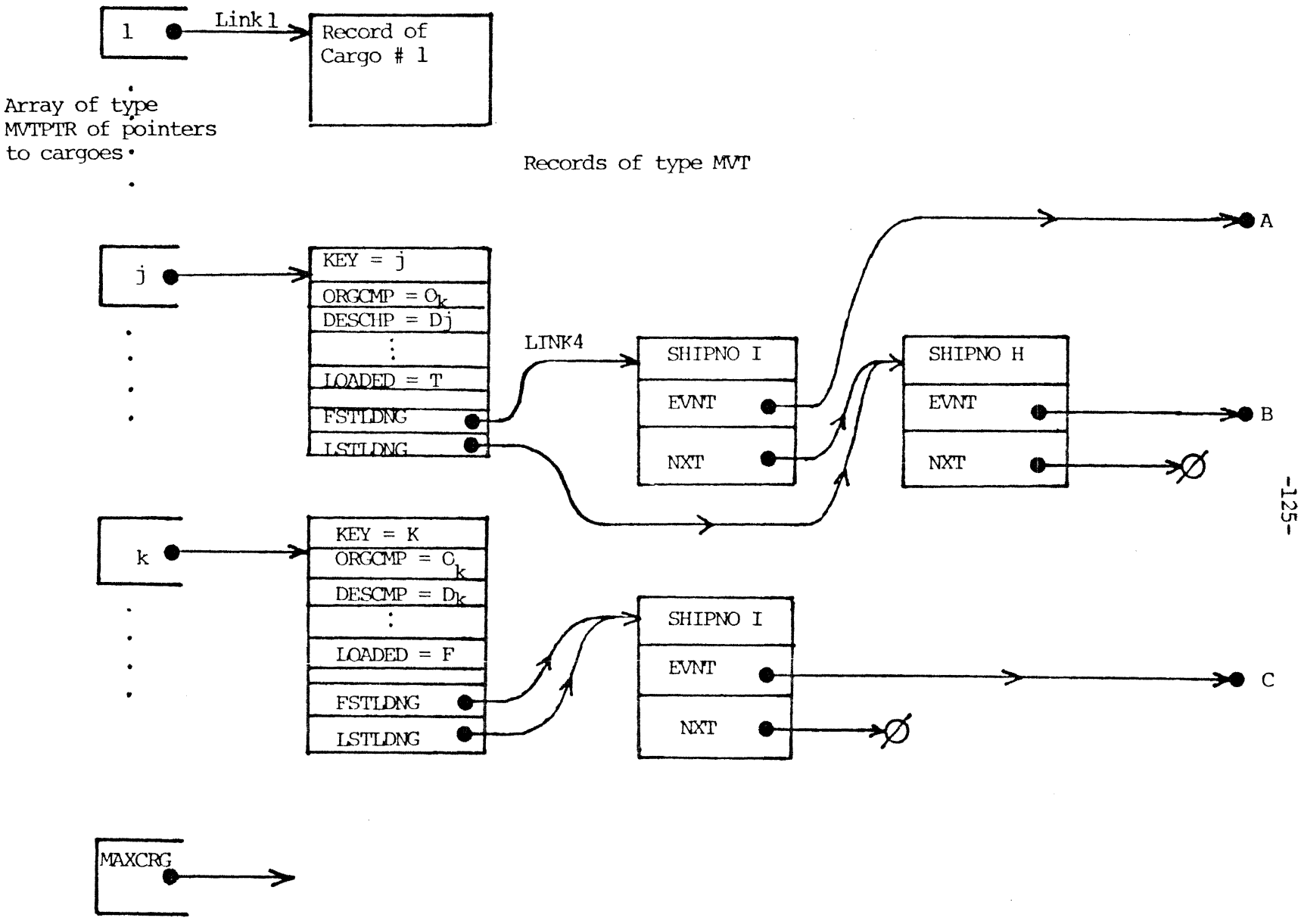


Figure A.2a: Cargoes and Ships Data Structures

(A, B, C, continued on Fig. A.2b)

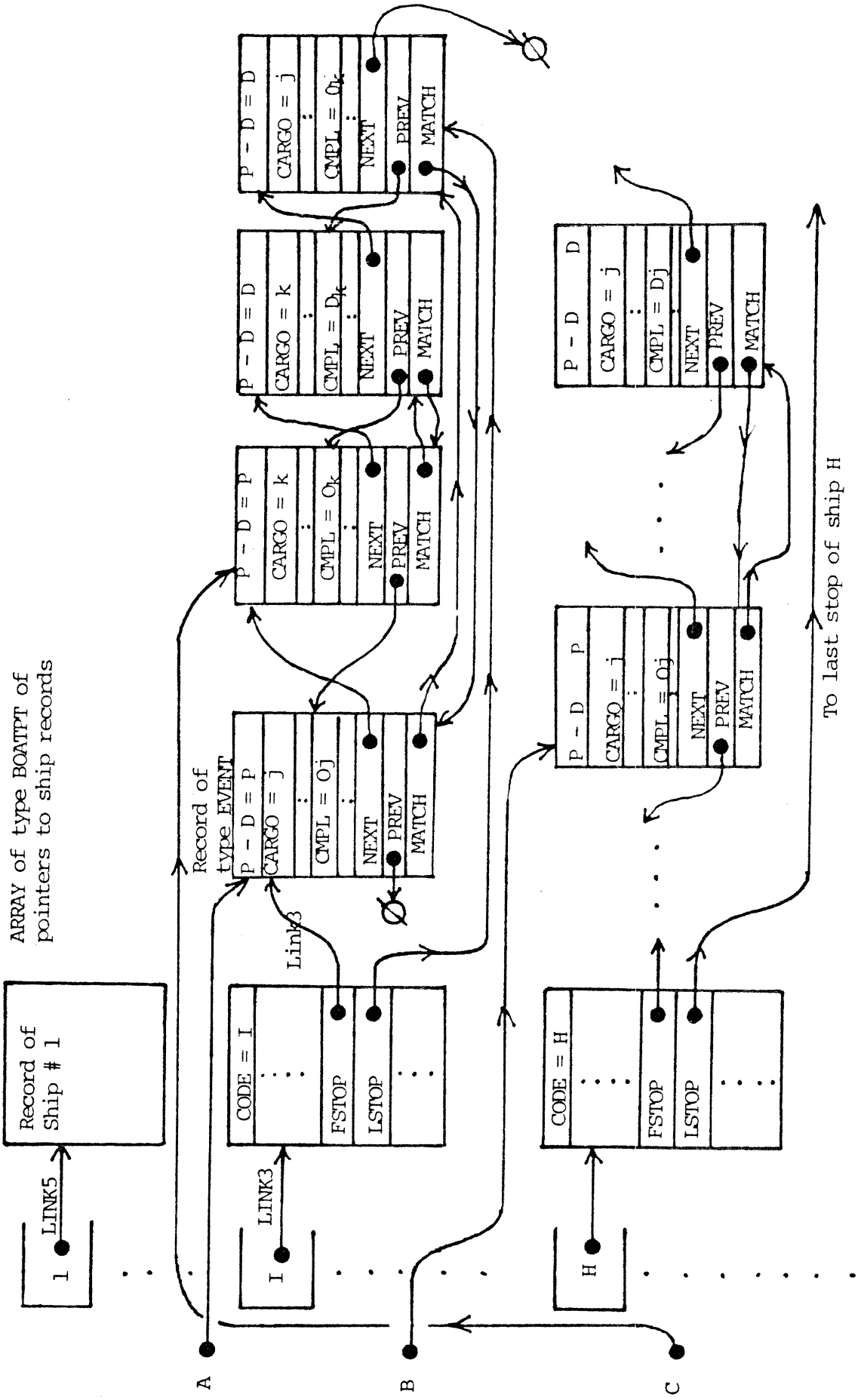


Figure A.2b: Cargoes and Ships Data Structures

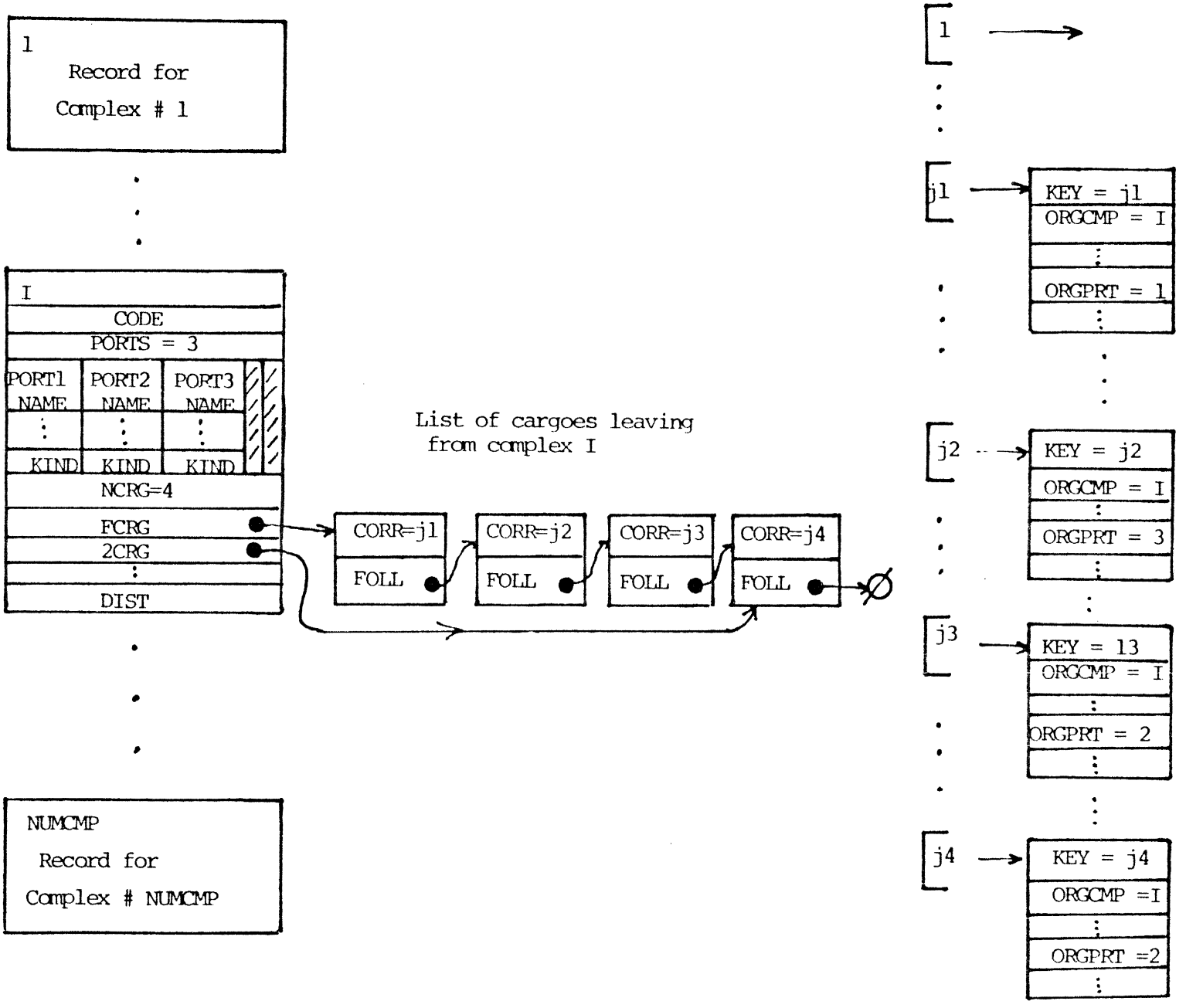


Figure A.3: Complexes, Ports and Cargoes

NV = 50

NU and NV are upper bounds on the number of ships and cargoes, respectively, in a given time window.

ARRSHP = ARRAY (1..NU) of integer

ARRCRG = ARRAY (1..NV) of integer

Arrays of type ARSHP and ARRCRG will carry the ships and cargoes in a given time window respectively (we refer to these arrays as cargostack and shipstack). In the present implementation, the array of ships will be constant (i.e. does not change with time windows).

BARR1 = ARRAY (1..NU) of boolean

After the optimization and assignment/deassignment in a time window, BARR1(j)=false if cargo j in cargostack was not assigned. This is used in setting up the next time window.

APPENDIX B

COMPUTER PROGRAM ORGANIZATION

B.1 Introduction

This appendix is devoted to giving a hierarchical overview of the computer programs in the MORSS package. This description does not include the preprocessing procedures, which (a) scan the complex and port files and create a single file containing all the data, and (b) lexicographically sort the cargoes according to preferred ship type, EPT, EDT, and LDT.

In general, the data is set up to match the needs of the data structures in MORSS (see Appendix A). Also, the very top program is a CMS exec called SCHED which sets up all files for reading and writing and actually starts MORSS.

Flow charts of the programs and subroutines of MORSS are shown in Figures B.2 to B.15. (all of which are at the end of this Appendix). Table B.1 provides an index for those figures and Figure B.1 displays some of the conventions used in the flowcharts. A description of the arguments of each subroutine is given in Appendix C.

Subroutines that merit special discussion are the following:

B.2 NETWORK Subroutine

NETWORK has 5 phases. In phase 1 NETWORK scans (simultaneously) the shipstack and the cargostack (as those were defined in Appendix A). For each cargo and ship pair, either the user or subroutine ASSNUTIL decide whether they are compatible, if they are, what the utility of the assignment is (as per Chapter 3). If the pair is compatible NETWORK will

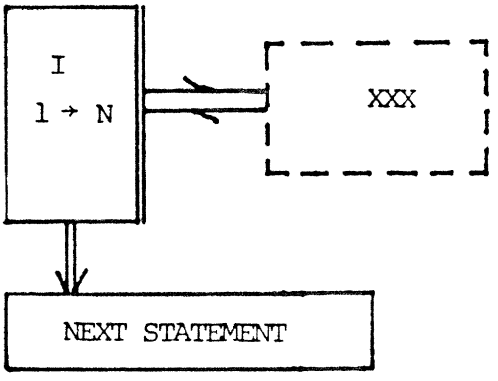
Program or Subroutine	Figure	Other Subroutines Called
MORSS	B.2	READIN, DISPLAY, SEEDS, SCHEDULE
READIN	B.3	-
SEEDS	B.4	ASSIGN
SCHEDULE	B.5	RLLTIME, NETWORK, PERMASSN
RLLTIME	B.6	SHIFT, DWN-LOAD
NETWORK	B.7.6	PEEKER, ASSNUTIL, FLOSUB, POSTOPT
PERMASSN	B.8	SHORT-SORT*, ASSNUTIL, ASSIGN
ASSIGN	B.9	UPD-P, QUPD
UPD-P	B.10	ADAPT*, UPD-CRG, INSERTS
DWN-LOAD	B.11	
QUPD	B.12	
SHIFT	B.13.2	
UPD-T	B.14.2	
UPD-CRG	B.15	

TABLE B.1

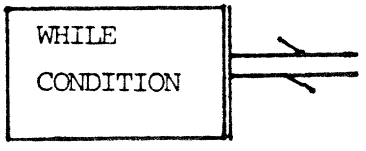
Index of MORSS Program and Routines

(*internal routine, not described here).

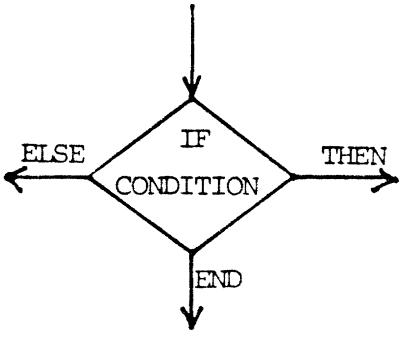
FLOWCHART CONVENTIONS



(DO Loop:
For I: = 1 to N DO
XXX
END
NEXT STATEMENT)



(WHILE Loop)



(IF Statement)

Figure B.1

set up an arc corresponding to the pair .

In phase 2 the arc data is consolidated to form a network (a bipartite graph) described by a forward-star data structure. In phases 1 and 2 NETWORK also sets up a "translation table" to make the nodes in the graph correspond to cargo and ship locations in the respective stacks.

In phase 3, the network data structure is written into a file.

In phase 4, the optimization algorithm (FLOSUB) is run. This algorithm prints the optimization results into a file (in general, we print the data into files for run control - in the ultimate version of MORSS, arguments should be passed directly between programs.)

Finally, in phase 5 subroutine POSTOPT reads in the optimization results, postprocesses them, and returns the assignemnt data to NETWORK. In spirit, the network is a bipartite graph (U,V,E) as shown in Figure B.7.1, where the set U corresponds to ships and the set V to cargoes. NETWORK sets up the graph so that every ship in the graph is connected to at least one cargo, and vice-versa. Thus, if a cargo in the cargo stack is incompatible with all ships, that cargo will not appear in the network. Also, the graph will contain a node corresponding to a "dummy ship" and a node corresponding to a "dummy cargo" to pick up extra supply or demand. Every real cargo is connected to the dummy ship and every real ship is connected to the dummy cargo. The nodes in the graph are labeled by numbers ranging from 0 to some number N+1, where N= number of real ships + number of real cargoes. 0 represents the dummy ship and N+1 the dummy cargo.

Example: Number of real ships=2, number of real cargoes=3. (see Figure B.7.2).

A forward-star representation of the graph is used. We keep the arrays:

(arc length) HEAD: heads of the arcs

(arc length) CST: cost of the arcs ordered as in HEADS

(node length) FIRST: index of first node (in HEAD) that node is connected to (0 if none)

(node length) LAST: index of last node (in HEAD) that node is connected to (0 if none)

(node length) RHS: supply of node. For real cargo nodes, it is -1. For real ship nodes, NETWORK sets it at 4 (at most 4 cargoes/ship). The dummy ship can have any RHS (we set it at 0) and the dummy cargo is given enough demand to balance those of real ship and cargo nodes.

We also keep

NUMARCS: total number of arcs in graph and use

SHIPS-IN-GRAPH: number of real ships in graph

CARGO-IN-GRAPH: number of real cargo in graph

Thus, for the previous example (see Fig. 8..7.2) we have:

HEAD = 3,4,3,5,3,4,5

CST = Large, Large,x,x,x,x,x (we shall talk about this later)

FIRST= 1,3,5,0,0,0

LAST = 2,4,7,0,0,0

RHS = 0,3,3,-1,-1,-4

Since not all ships or cargoes in their stacks may appear in the graph, the node number assigned to (say) a given cargo may be different from that cargo's position in the cargo stack. To provide translation, we use the arrays EQUIV and BACK. Given a node $j \geq 1$, EQUIV(j) will be the

position of

$$\begin{cases} \text{the ship corresponding to } j, \text{ if } j \leq \text{SHIPS-IN-GRAPH} \\ \text{the cargo corresponding to } j, \text{ if } j > \text{SHIPS-IN-GRAPH} \end{cases}$$

in the shipstack or cargostack, respectively.

For instance, if SHIPS-IN-GRAPH = 6 and EQUIV(2) = 4, node 2 in the graph will be the 4th ship in the shipstack.

BACK is the inverse function of EQUIV.

As mentioned in Appendix A, the constants NU (=50) and NV (=50) are upper bounds, respectively, to the number of ships and cargoes in their stacks. NETWORK initially assumes that indeed there are NU ships and NV cargoes in the graph. Later this overestimate is corrected and in the same process the final network preresentation is obtained. We shall refer to this process as the "contraction". The initial overestimate is also reflected in errays EQUIV and BACK, as well as in the variable MARK (= number of arcs in graph). Initially, MARK is set at NV (NETWORK assumes that all cargoes are in the graph, so node 0 - the dummy ship - is connected to NV nodes.)

In order to know, at any point before the contraction, which ships and cargoes are already in the graph, we use the array YES of boolean values. For $i \leq$ NL

YES(i) = T if i^{th} ship in shipstack is in graph yet

F otherwise

Similarly, for $1 \leq j \leq$ NV

T if j^{th} cargo in cargostack is in graph yet

YES(j + NU) =

F otherwise

YES is initialized F for all entries.

The main work done before the contraction goes as follows: We enumerate the shipstack, i.e. we scan STACKSHP(I) for $1 \leq I \leq$ SHPNUZ

Given I, we enumerate the cargostack, i.e. we scan STACKCRG(J) for $1 \leq J \leq$ CRGC.

Suppose, that given I and J, say by calling ASSNUTL we determine that the pair is compatible. Then: First, we increase MARK by one (an extra arc in graph). We check to see whether YES (J + NU) = T. Suppose it is not. That means that the J^{th} cargo has not been found compatible to any previous ship or that this is the first time we scan cargo J. So we:

set YES (J + NU) = T (cargo is now in graph)

set EQUIV (CARGOS-IN-GRAPH + NU) = J (temporarily,

node number CARGOS-IN-GRAPH + NU corresponds to the

J^{th} cargo)

set BACK (J + NU) = CARGOS-IN-GRAPH + NU

We also check the value of YES(I), and perform similar operations as above. In addition, if YES(I) was F when checking, arc (I,J) is the first arc starting from ship I, so we also set FIRST (SHIP-IN-GRAPH) = MARK (ship I will be the SHIPS-IN-GRAPH th node)

Finally, since I,J are compatible, we also do:

HEAD (MARK) = BACK (J + NU) (set head of new arc)

After enumerating all cargoes, given ship I, we check the value of YES(I). If = F, then we know the ship I was found incompatible to all cargoes, so it should not be in the graph at all. On the other hand, if YES (I) = T, then ship I should be connected to the dummy cargo (temporarily represented by node # -1). So we go:

MARK = MARK + 1 (one more arc)

LAST (BACK (I)) = MARK (BACK I is the node that represents ship I. Arc # MARK is the last arc starting from ship I)

HEAD (MARK) = -1

Once the double loop (ships and cargoes) is finished, the contraction begins. First of all, node 0 (the dummy ship) will be connected only to CARGOS-IN-GRAPH nodes, not NV nodes, so we go FIRST(0) = 1, LAST (0) = CARGOS-IN-GRAPH and for $1 \leq J \leq$ CARGOS-IN-GRAPH+J (The lowest numbers node corresponding to a cargo will be node # SHIPS-IN-GRAPH+1).

and EQUIV (J + SHIPS-IN-GRAPH) = EQUIV (J + NU)

This says that the node previously numbered J + NU is now numbered J + SHIPS-IN-GRAPH (remember that the EQUIV array tells us what cargo the node represents).

Similarily, BACK (EQUIV (J + NU) + NU) = J + SHIPS-IN-GRAPH

An example of the arrays before and after the contraction, for NU=NV=5 will help clarify the situation (see Figure B.7.3).

We must also contract the HEAD array and reset the FIRST and LAST errays. To contract HEAD, we first note that the number of ships in the

graph was initially overestimated by CORRTERM1 = NU -SHIPS-IN-GRAPH. So, if an entry in the HEAD array is

= -1 (dummy cargo)
= -1 (real cargo)

} it should become {

1+ SHIPS-IN-GRAPH + CARGO-IN-GRAPH

<u>decreased</u> by CORRTERM

In addition, the original number of cargoes in the graph was overestimated by CORRTERM2 = NV-CARGOS-IN-GRAPH. Thus the entire HEAD array should be shifted left by CORRTERM2, and similarily, for $1 \leq I \leq$ SHIPS-IN-GRAPH, FIRST (I) and LAST (I) should be decreased by CORRTERM2. An example, before and after the contraction is shown in Figure B.7.4. The reader may verify that the graph in question is the one shown in Figure B.7.5.

To finish the contraction, we must set FIRST=LAST=0 for nodes corresponding to cargoes, and also take care of contracting the CST array, setting the RHS array and a few similar things. This will set up the network, which is printed in a file by writing, in the given order

NS(=SHIPS-IN-GRAPH + 1 = number of sources in graph)
CARGOS-IN-GRAPH (= number of sinks)
NUMARCS (= LAST (SHIPS-IN-GRAPH) = number of arcs)
array FIRST
array LAST
array RHS
array HEAD

array CST

Finally, NETWORK calls FLOSUB and next POSTOPT.

The flowchart for NETWORK is given in a highly abbreviated form, in Figure B.7.6.

B.2 SHIFT Subroutine

"Parallel" to the cargostack, we keep an array of boolean entries (one entry per cargo in the cargostack). After assignments and deassignments, the entries in this array will be T or F corresponding to assigned or nonassigned cargoes respectively. SHIFT looks only at the F-cargoes, and shifts all of those to the beginning of the stack. An example is shown in Figure B.13.1.

In addition, SHIFT utilizes two parameters: CRGC and LAST. CRGC is the total number of cargoes in the stack. The cargoes between LAST+1 and CRFC are always automatically deassigned (last portion of time window). Finally, F-cargoes are shifted only when they have not been attempted to be assigned more than a fixed number of attempts (NATTB).

B.3 UPD-T Subroutine

This procedure creates a dummy schedule for a given ship, in order to test the utility of a certain cargo insertion. There are 4 main pieces of data involved, but before describing these, a general overview is given.

Suppose the ships schedule looks like the one shown in Figure B.14.1(a), where the S_i's are the stops. Suppose we want to insert the pickup P of the cargo between S_1 and S_2; and the delivery D between S_3 and S_4. Then UPD-T will first create a schedule that looks like the one shown

in Figure B.14.1(b).

Stop S_1 and the old schedule are not lost, rather the new schedule is simulated by using special pointers and the old stops.

The first data structure is used precisely to achieve this. In the program it is called NUTOREC and consists of an array of pointers of type Link3. NUTOREC(1) points to the simulated pickup (as above). In the example above, also, we have NUTOREC(4) pointing to the simulated delivery.

After the insertion of the two stops, some of the data contained in the old stop records will be incorrect (with respect to the simulated schedule), for instance the time data. Rather than change the data in the old stops we keep 2 arrays, NUTIME and NUSLK, which contain the updated times and slacks for the simulated schedule (in both cases the arrays start with the new pickup). Finally, array OUTSLK contains the "future slacks". All the arrays have COUNT entries. In the example above first COUNT=4, and after inserting the delivery, COUNT=5.

There are three routines that are used by UPD-T which will not be described here in detail. They are SHIFT (internal, not to be confused with procedure SHIFT, described earlier), TRAVERST and SLAXER. TRAVERSE takes the original schedule of the ship and generates the updated schedule containing the pickup only. SHIFT adds the delivery to the updated schedule. SLAXER takes this schedule and computes the array OUTSLK.

When we insert the delivery, the variable PLACE indicates where in the updated schedule the delivery will be (counted from the pickup). For instance, in the example above, PLACE=4. Before, a pointer of type Link3 points to the stop before which we are inserting the stop (whether pickup or delivery).

As a delivered cargo is inserted, the old arrays containing time slack and stops are preserved in arrays PTIME, PSLK and PTOREC so that they can be used for the next delivery attempt.

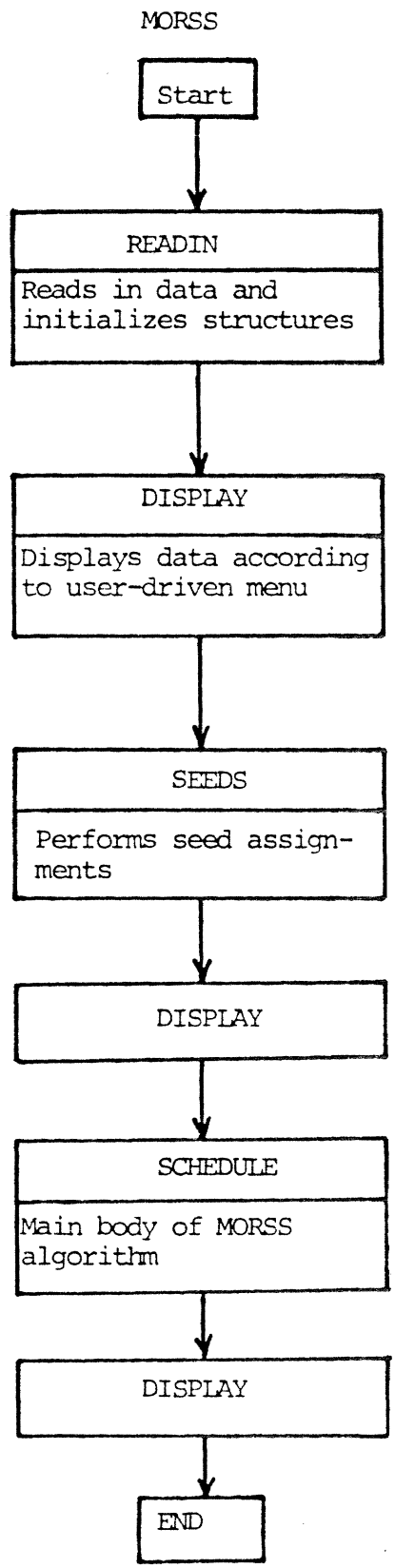

MORSS
Start
READIN
Reads in data and initializes structures
DISPLAY
Displays data according to user-driven menu
SEEDS
Performs seed assign-ments
DISPLAY
SCHEDULE
Main body of MORSS algorithm
DISPLAY
END

Figure B.2

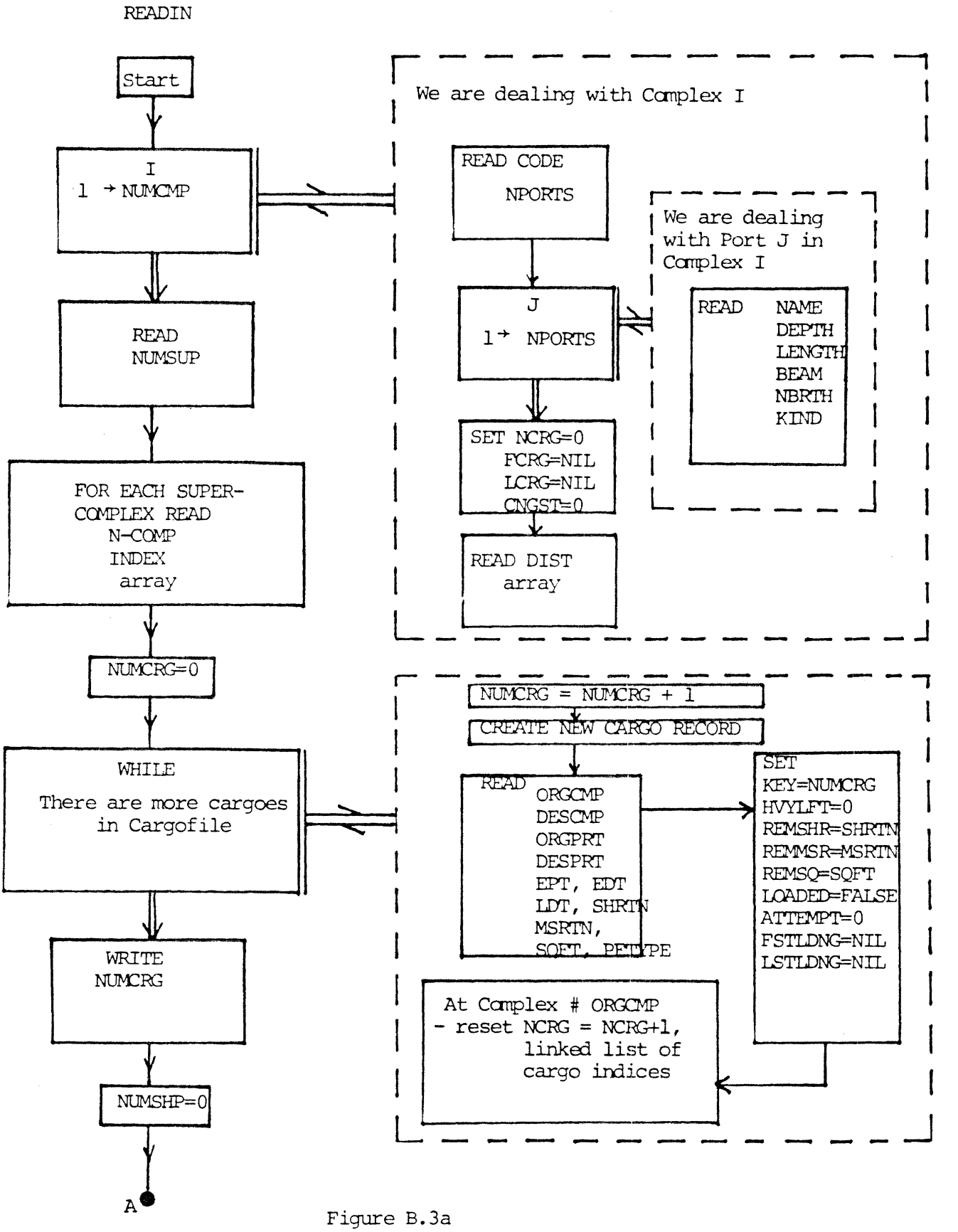
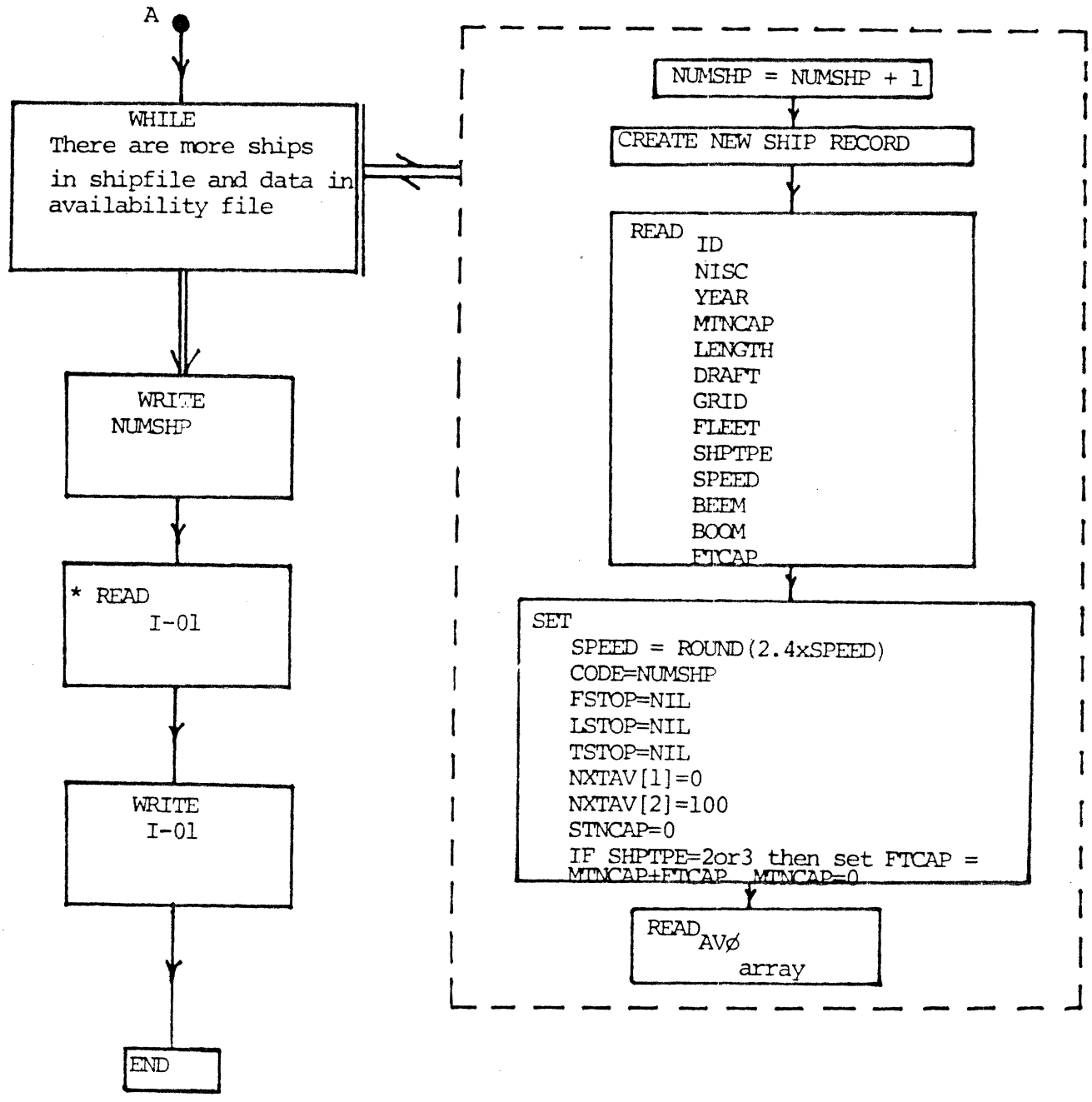


Figure B.3a

(A: continued in Fig. B.3b)



* I-01, user set, controls how much output is generated by MORSS

Figure B.3b

SEEDS

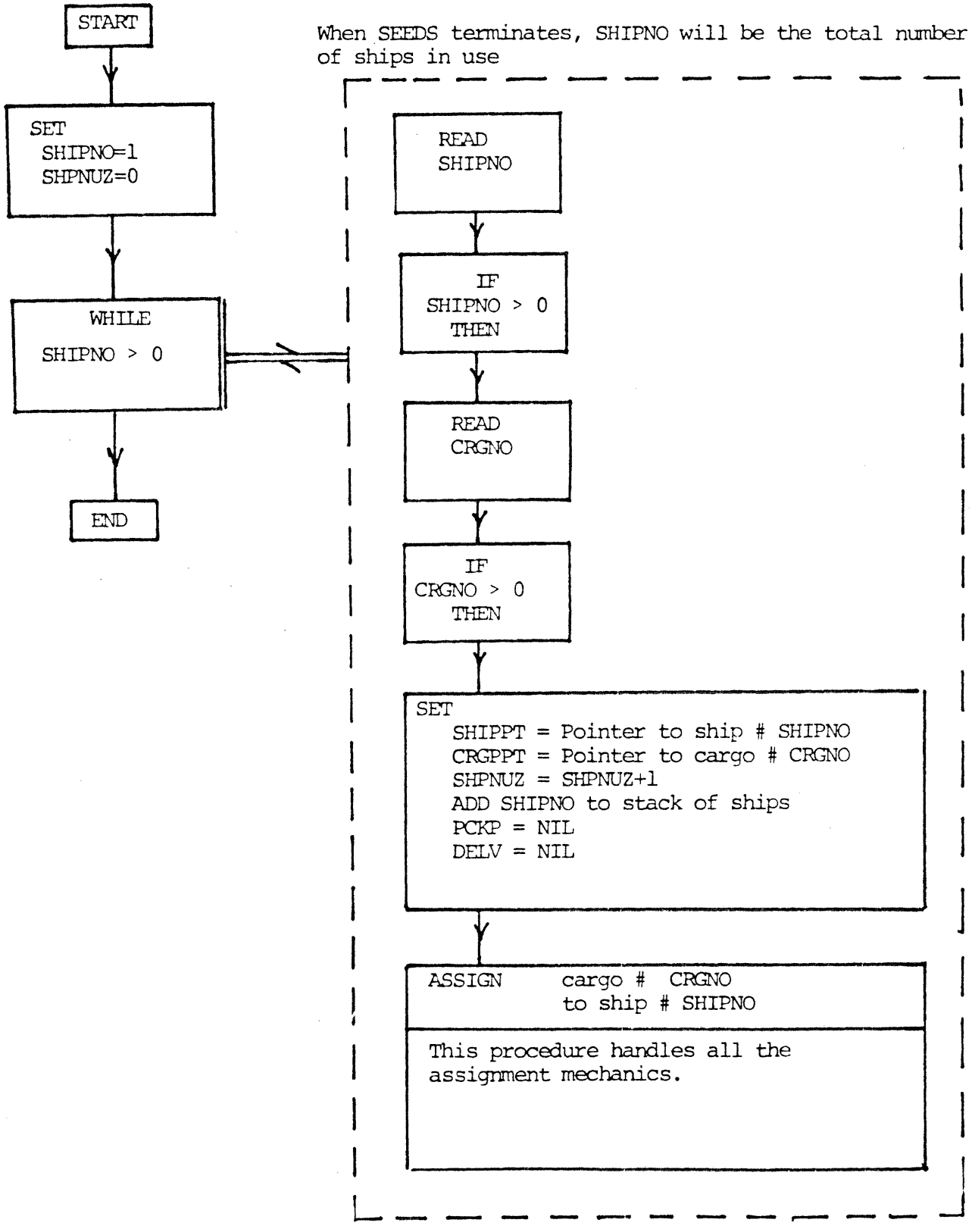


Figure B.4

SCHEDULE

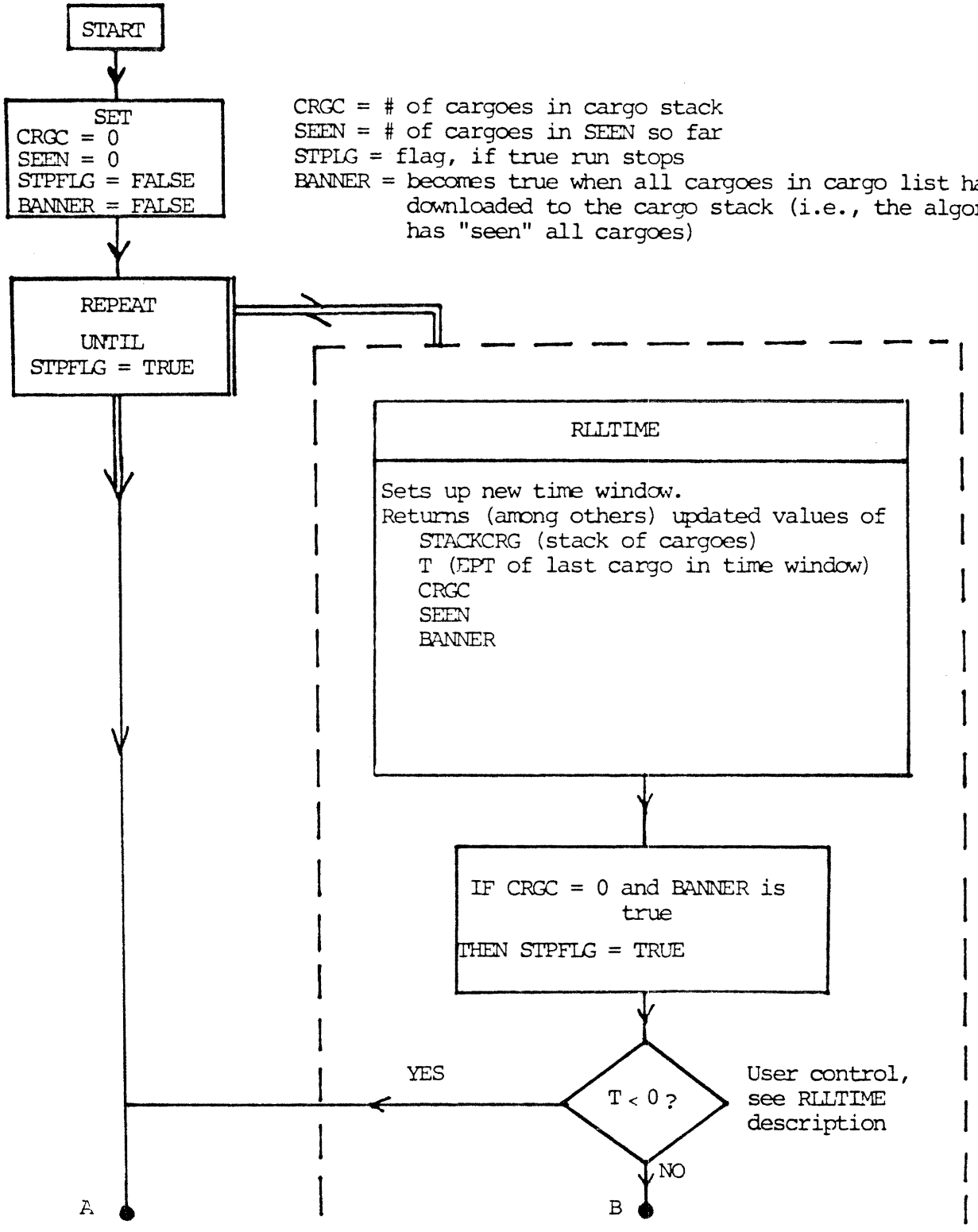


Figure B.5.a

(A, B: cont'd. in Figure B.5.b)

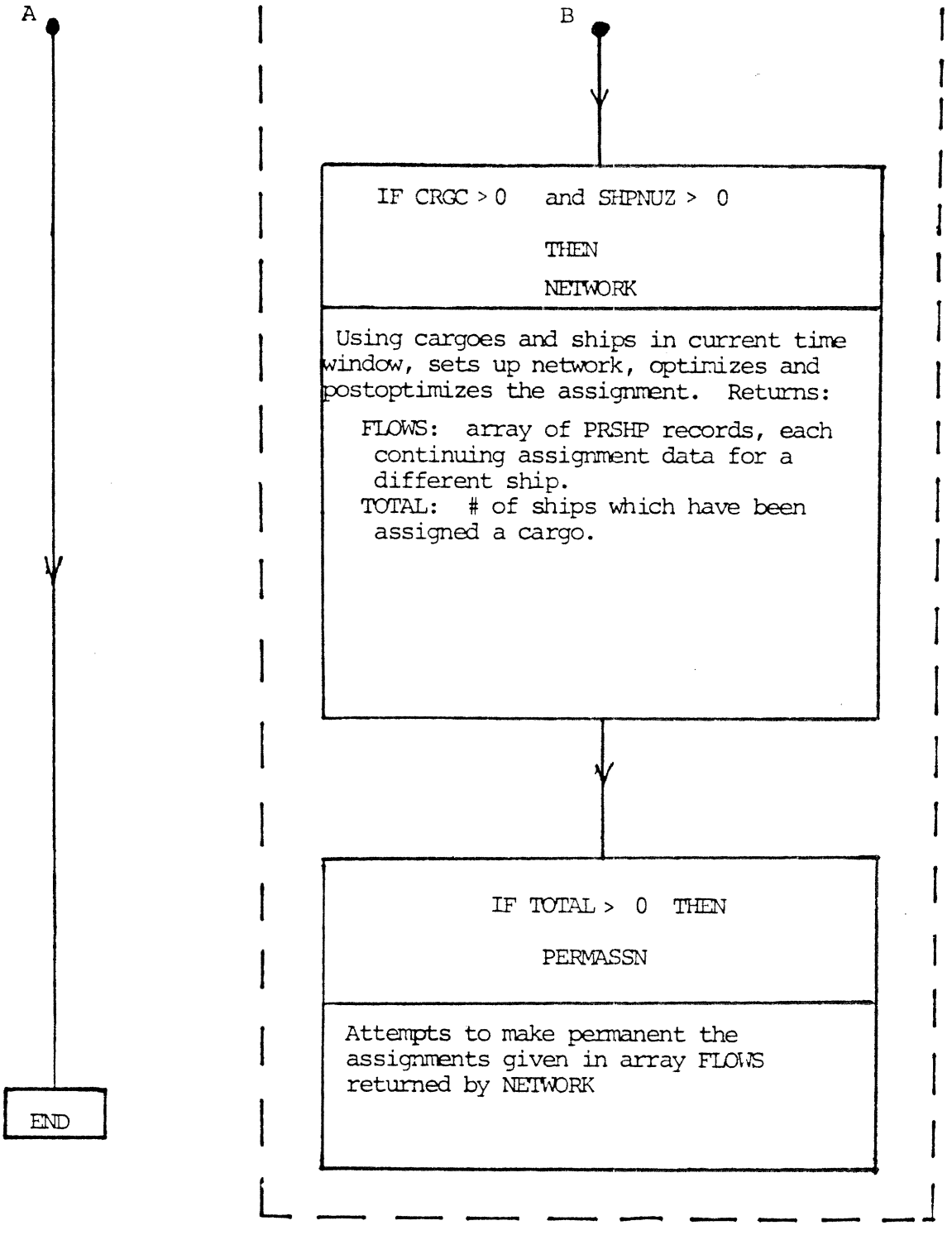

A
B
IF CRGC > 0 and SHPNUZ > 0
THEN
NETWORK
Using cargoes and ships in current time window, sets up network, optimizes and postoptimizes the assignment. Returns:
FLOWS: array of PRSHP records, each continuing assignment data for a different ship.
TOTAL: # of ships which have been assigned a cargo.
IF TOTAL > 0 THEN
PERMASSN
Attempts to make permanent the assignments given in array FLOWS returned by NETWORK
END

Figure B.5.b

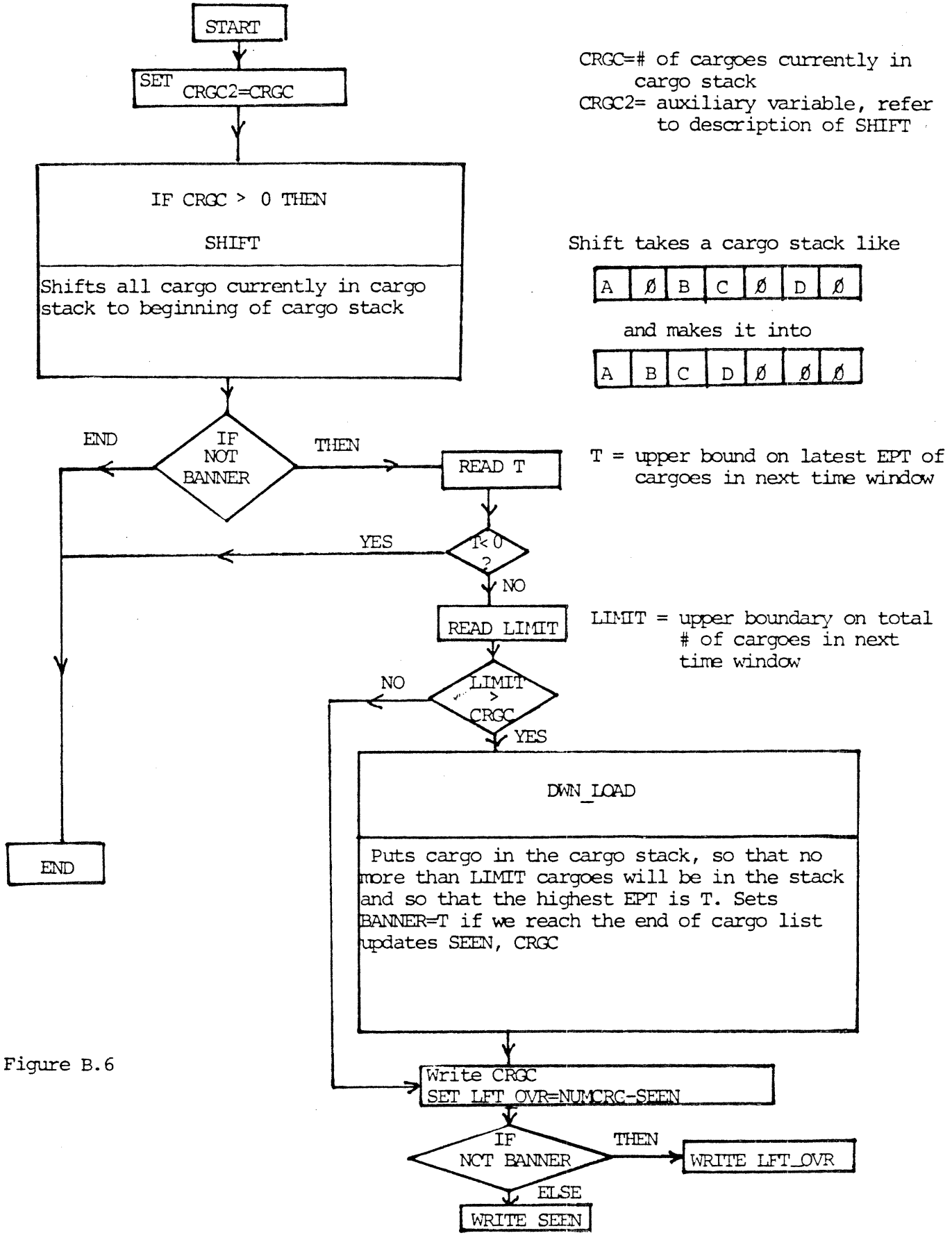

START
SET CRGC2=CRGC
IF CRGC > 0 THEN
SHIFT
Shifts all cargo currently in cargo stack to beginning of cargo stack
CRGC=# of cargoes currently in cargo stack
CRGC2= auxiliary variable, refer to description of SHIFT
Shift takes a cargo stack like
A Ø B C Ø D Ø
and makes it into
A B C D Ø Ø Ø
END
IF NOT BANNER
THEN
READ T
T = upper bound on latest EPT of cargoes in next time window
YES
T< 0 ?
NO
READ LIMIT
LIMIT = upper boundary on total # of cargoes in next time window
NO
LIMIT > CRGC
YES
END
DWN_LOAD
Puts cargo in the cargo stack, so that no more than LIMIT cargoes will be in the stack and so that the highest EPT is T. Sets BANNER=T if we reach the end of cargo list updates SEEN, CRGC
Write CRGC
SET LFT_OVR=NUMCRC-SEEN
IF NOT BANNER
THEN
WRITE LFT_OVR
ELSE
WRITE SEEN

Figure B.6

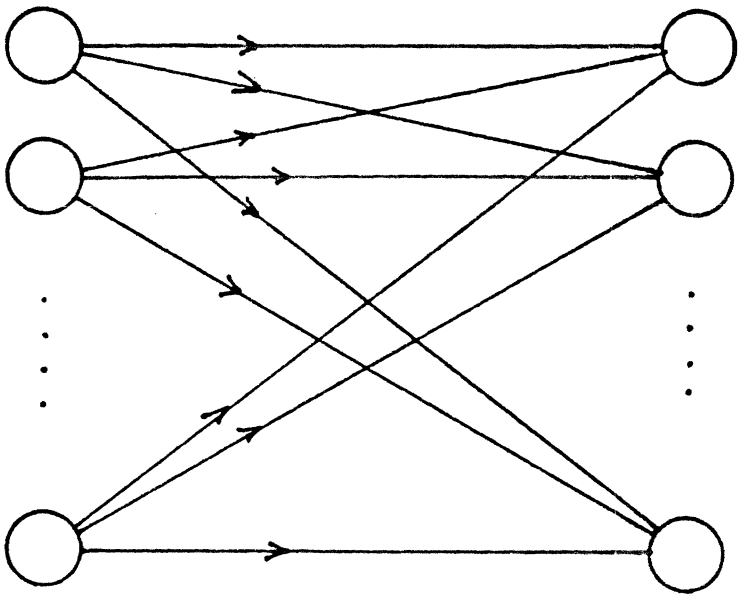


Figure B.7.1

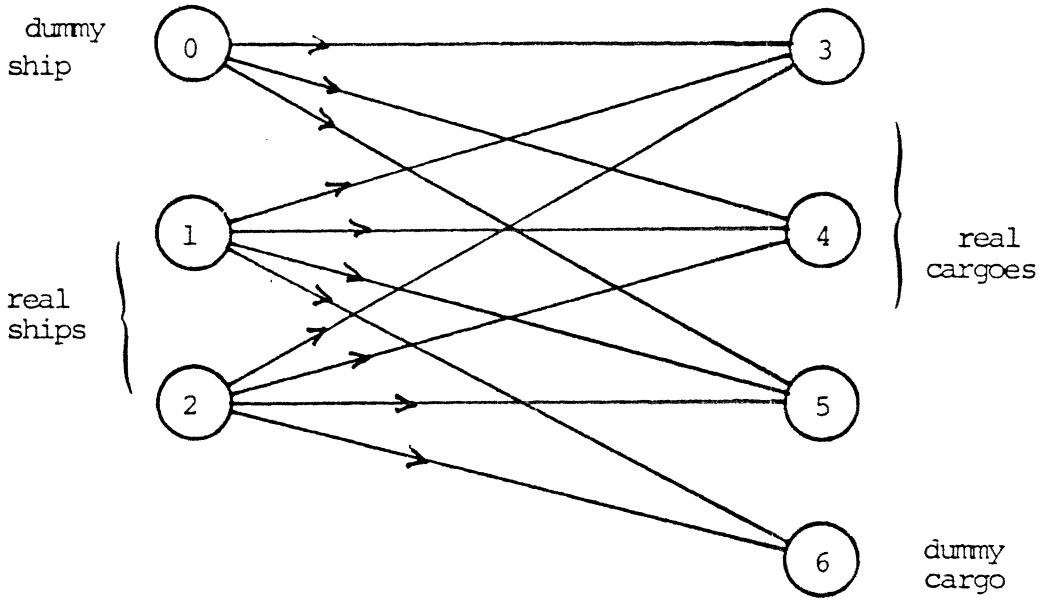
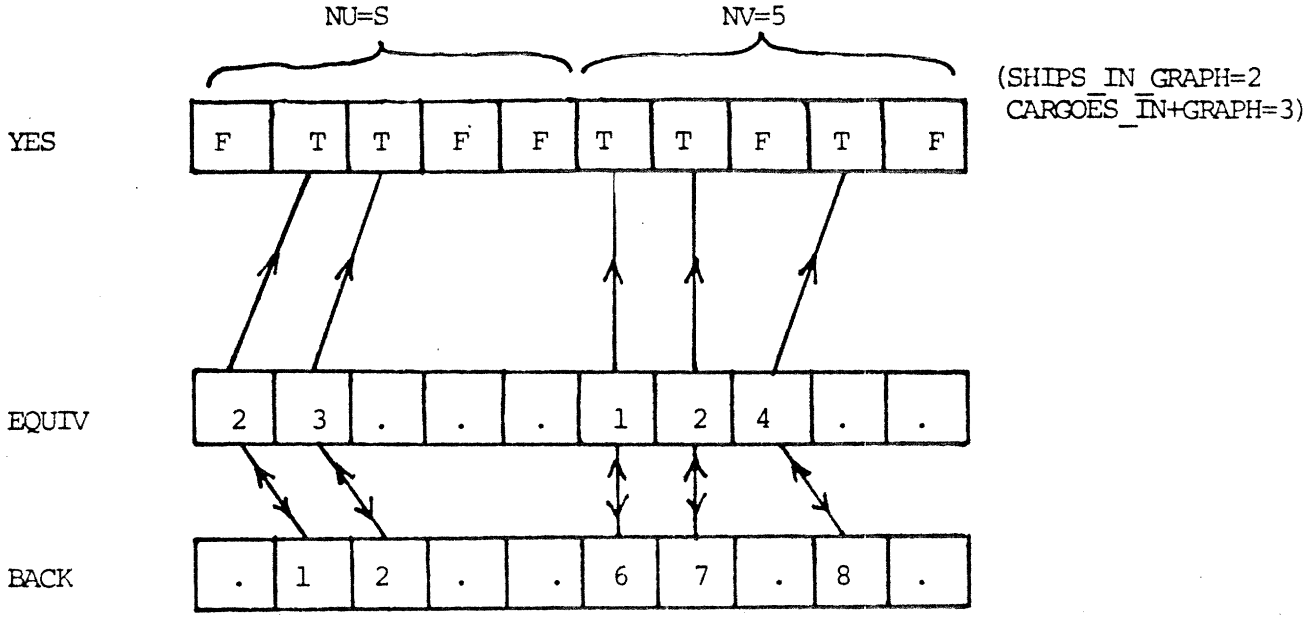


Figure B.7.2

BEFORE CONTRACTION:



AFTER CONTRACTION:

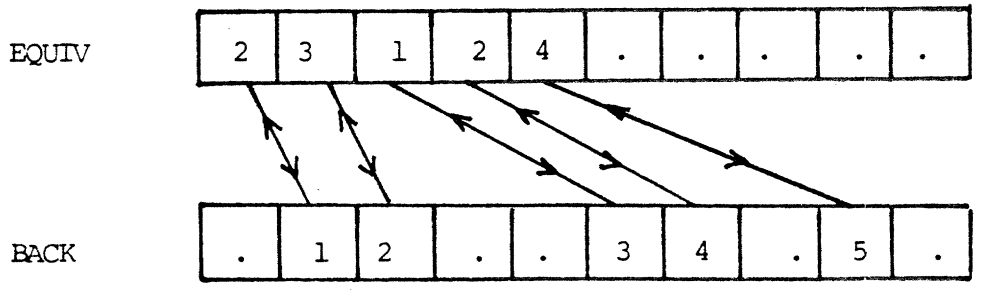
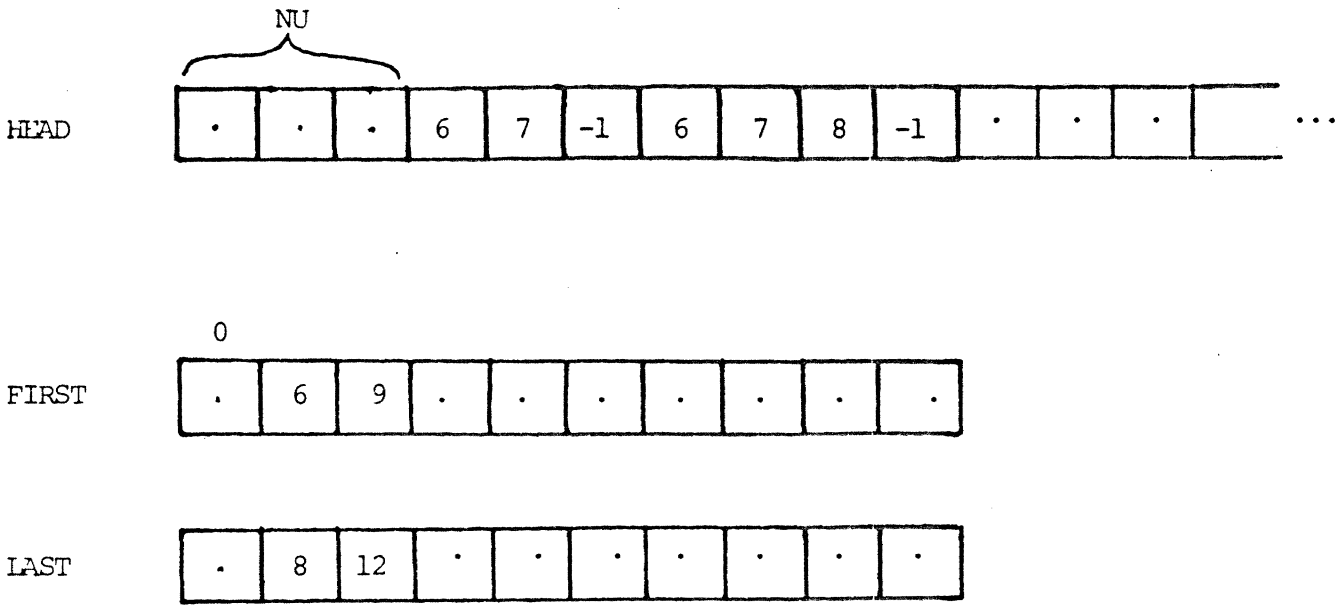


Figure B.7.3

BEFORE CONTRACTION:



AFTER CONTRACTION:

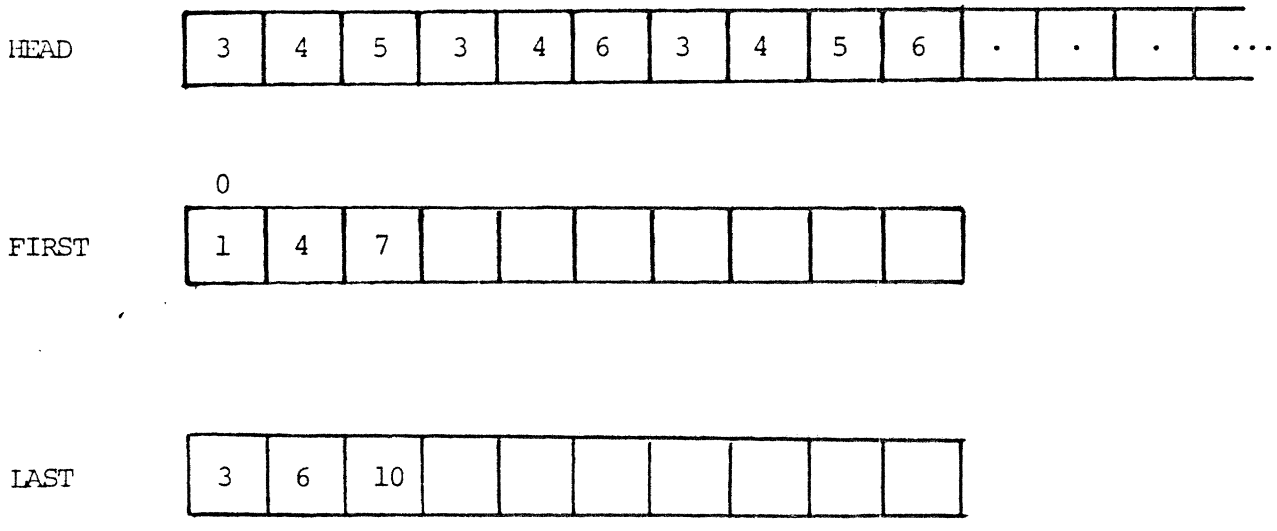


Figure B.7.4

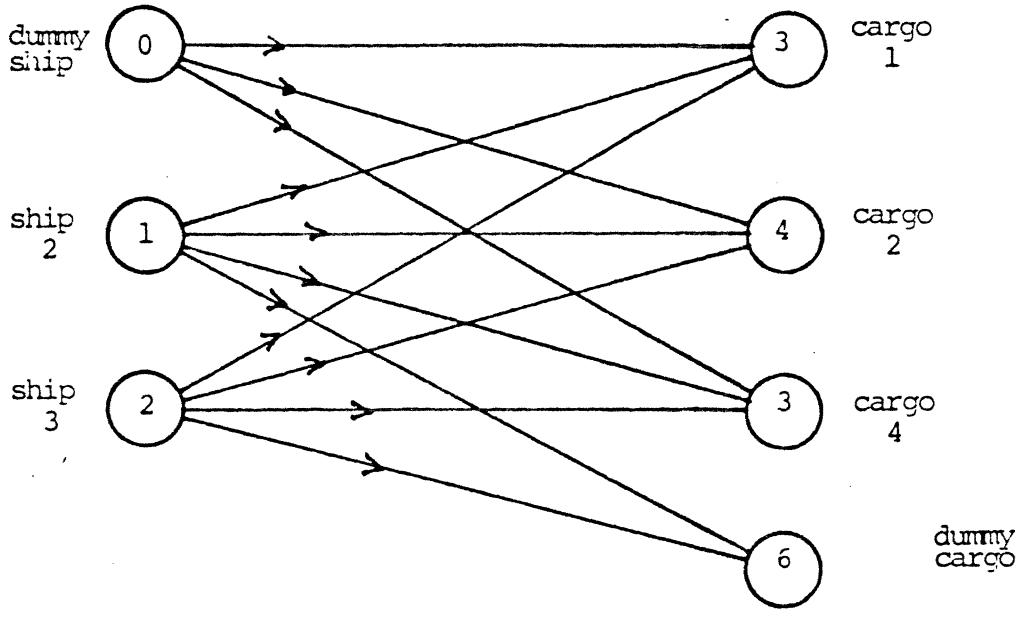

dummy ship
0
ship 2
1
ship 3
2
3
4
3
6
cargo 1
cargo 2
cargo 4
dummy cargo

Figure B.7.5

NETWORK

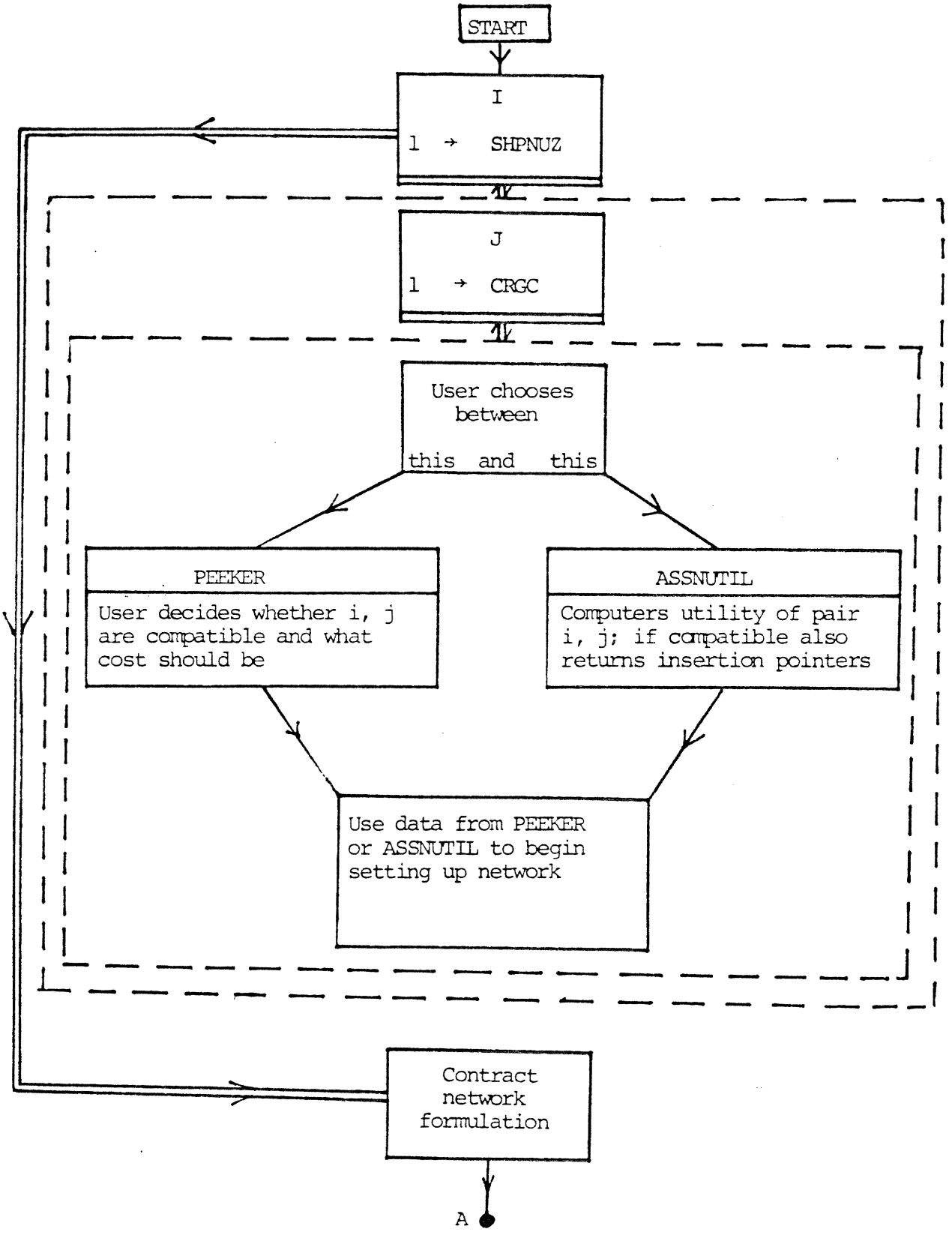


Figure B.7.6a
(A: cont'd in Figure B.7.6b)

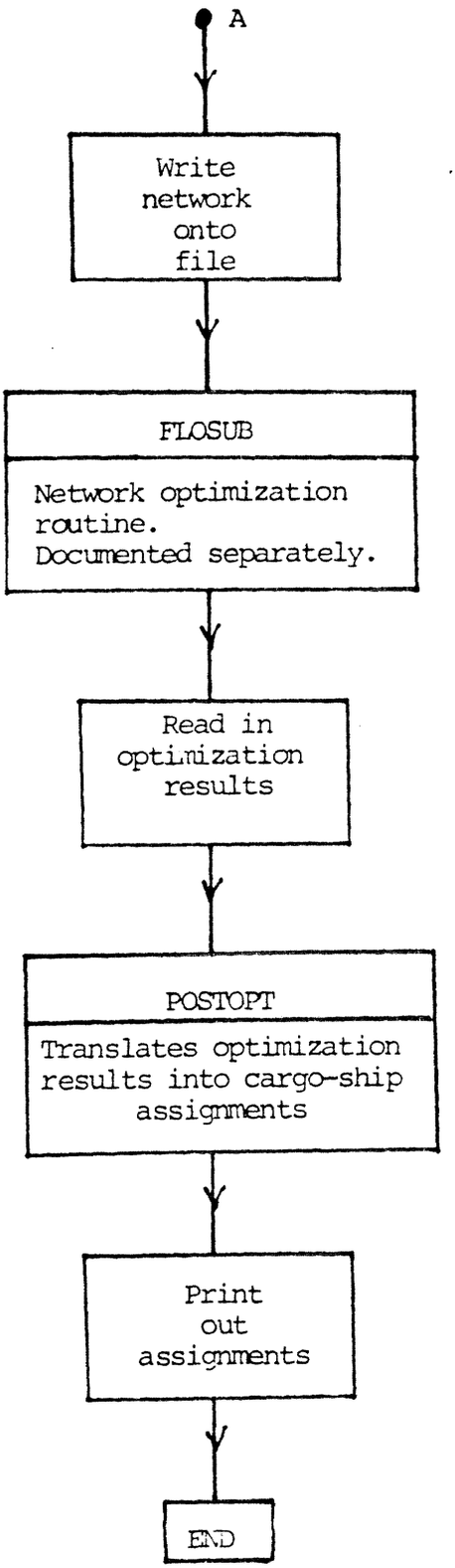

A
Write network onto file
FLOSUB
Network optimization routine. Documented separately.
Read in optimization results
POSTOPT
Translates optimization results into cargo-ship assignments
Print out assignments
END

Figure B.5.6b

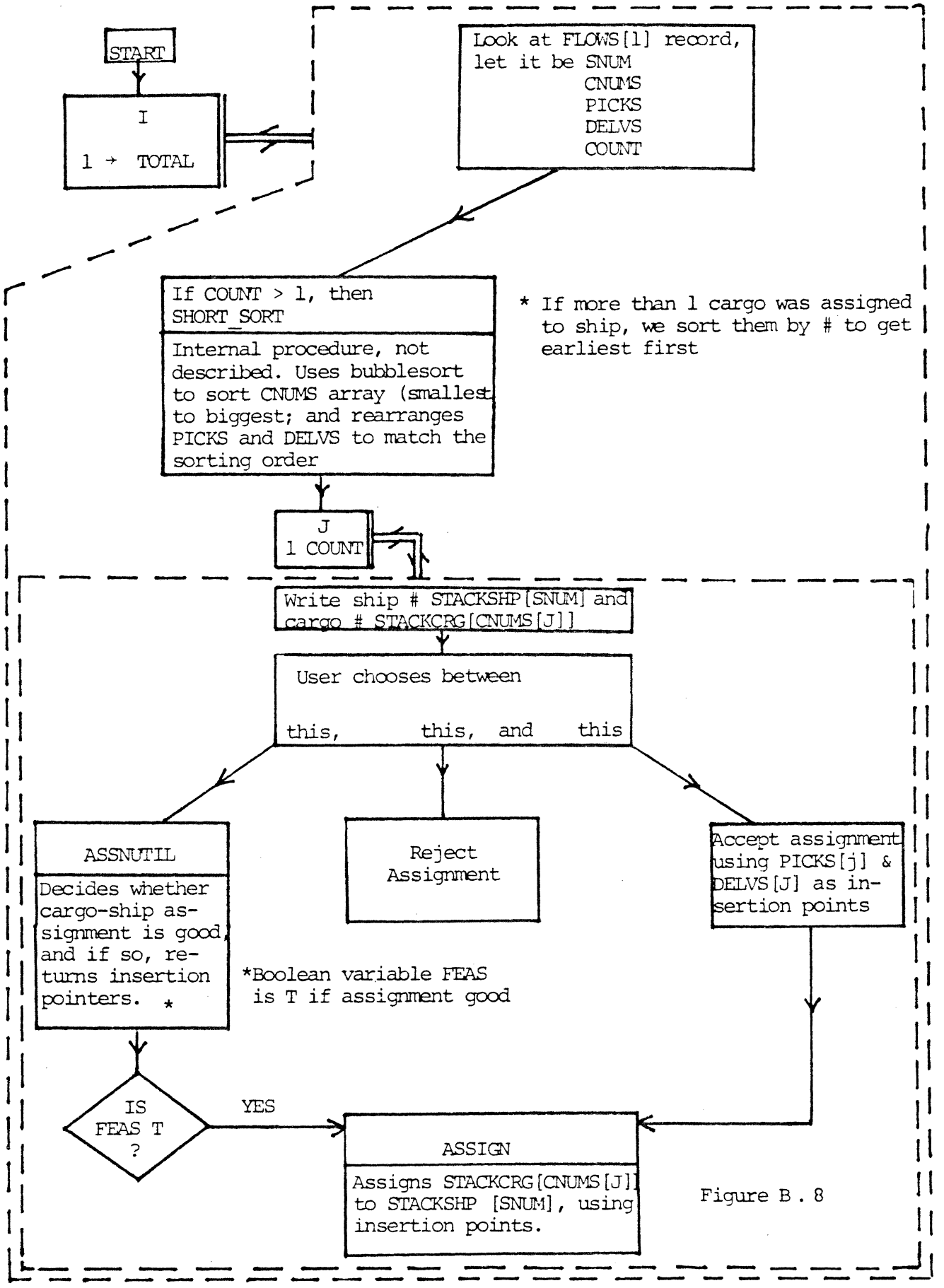

START
I
1 → TOTAL
Look at FLOWS[1] record, let it be SNUM
CNUMS
PICKS
DELVS
COUNT
If COUNT > 1, then SHORT_SORT
Internal procedure, not described. Uses bubblesort to sort CNUMS array (smallest to biggest; and rearranges PICKS and DELVS to match the sorting order
* If more than 1 cargo was assigned to ship, we sort them by # to get earliest first
J
1 COUNT
Write ship # STACKSHP[SNUM] and cargo # STACKCRG[CNUMS[J]]
User chooses between
this, this, and this
ASSNUTIL
Decides whether cargo-ship assignment is good, and if so, returns insertion pointers. *
Reject Assignment
Accept assignment using PICKS[j] & DELVS[J] as insertion points
*Boolean variable FEAS is T if assignment good
IS FEAS T ?
YES
ASSIGN
Assigns STACKCRG[CNUMS[J]] to STACKSHP [SNUM], using insertion points.

Figure B . 8

ASSIGN

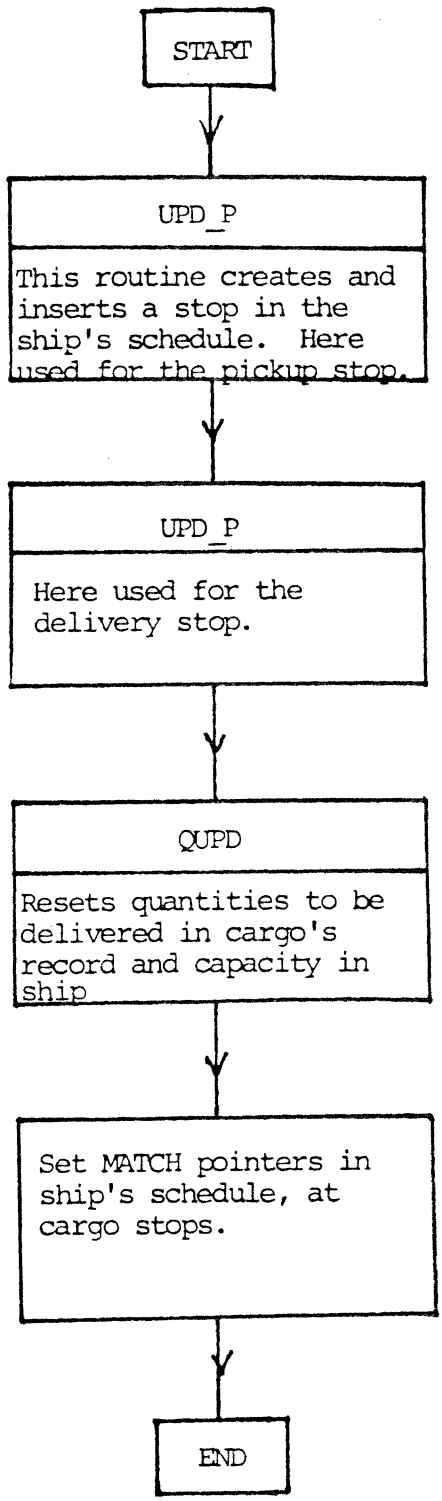


Figure B.9

UPD_P

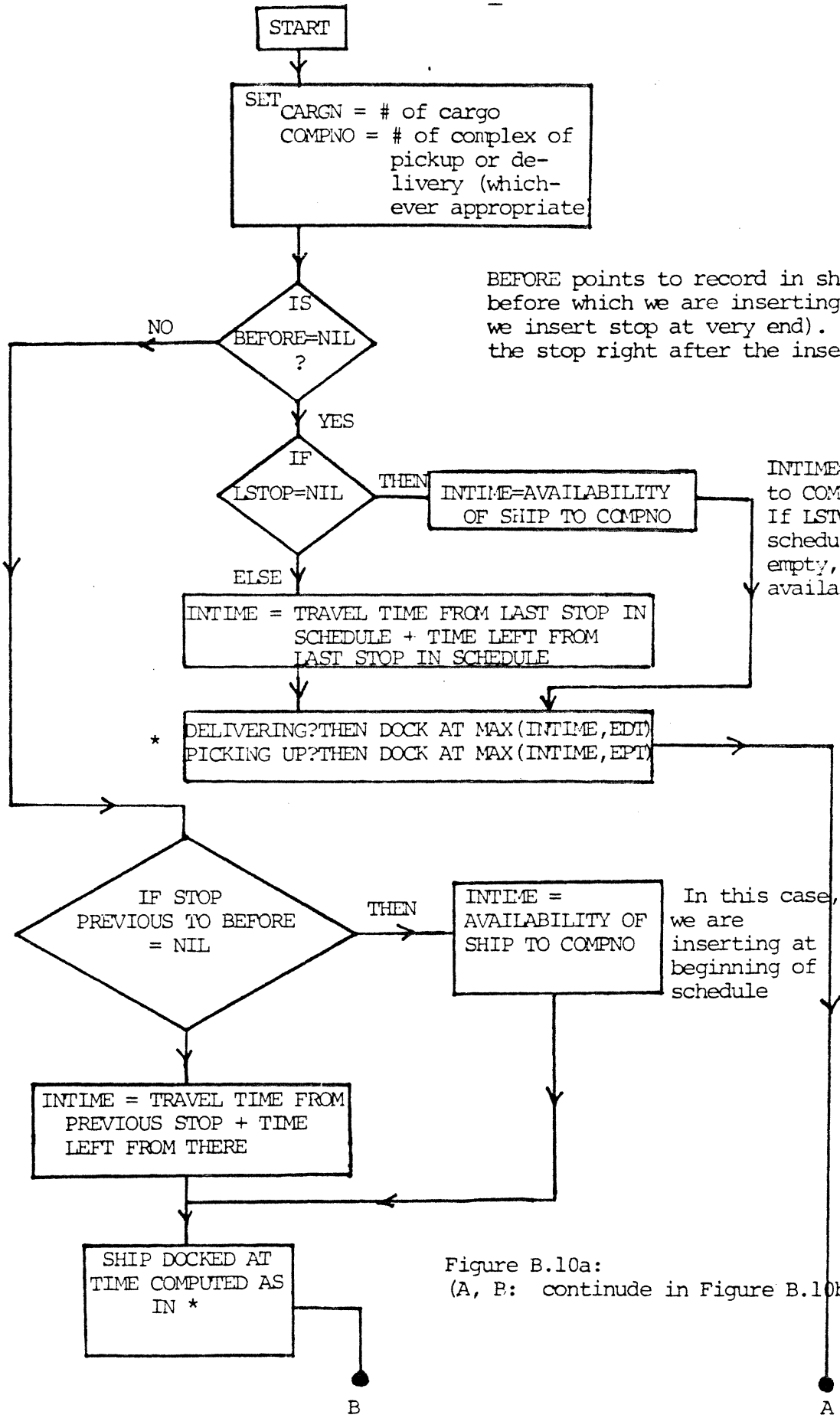


Figure B.10a:
(A, B: continude in Figure B.10b)

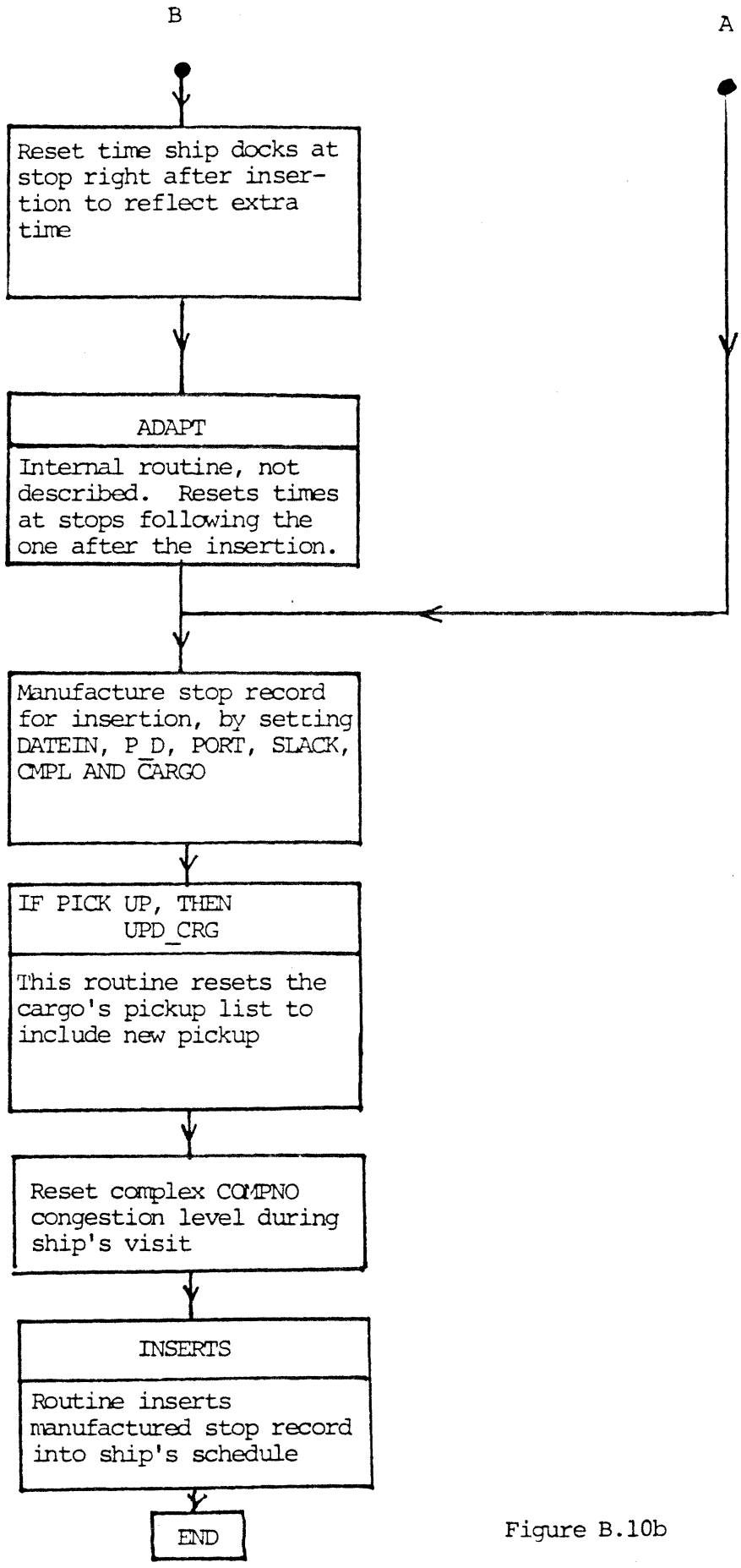

B
A
Reset time ship docks at stop right after insertion to reflect extra time
ADAPT
Internal routine, not described. Resets times at stops following the one after the insertion.
Manufacture stop record for insertion, by setting DATEIN, P_D, PORT, SLACK, CMPL AND CARGO
IF PICK UP, THEN UPD_CRG
This routine resets the cargo's pickup list to include new pickup
Reset complex COMPNO congestion level during ship's visit
INSERTS
Routine inserts manufactured stop record into ship's schedule
END

Figure B.10b

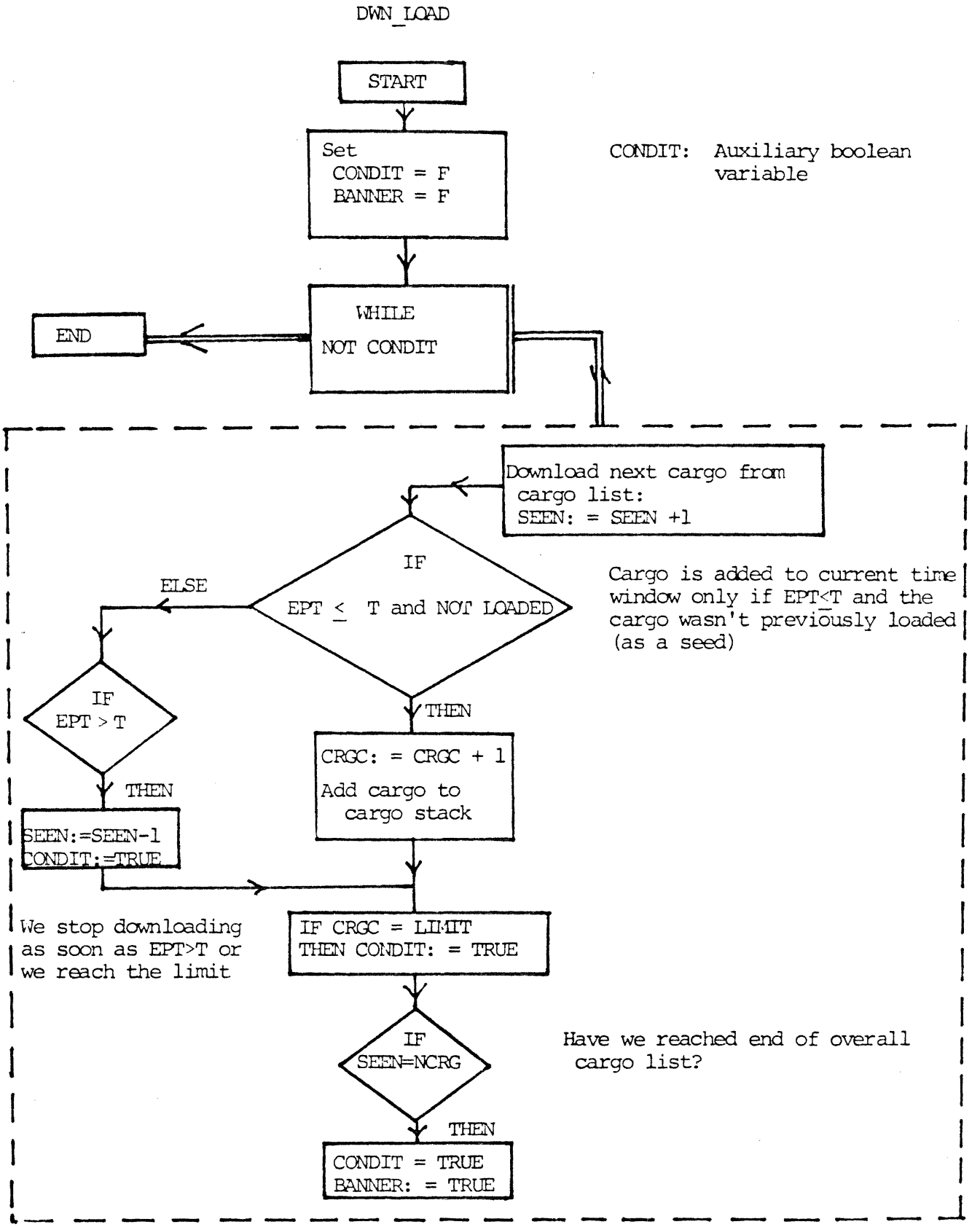
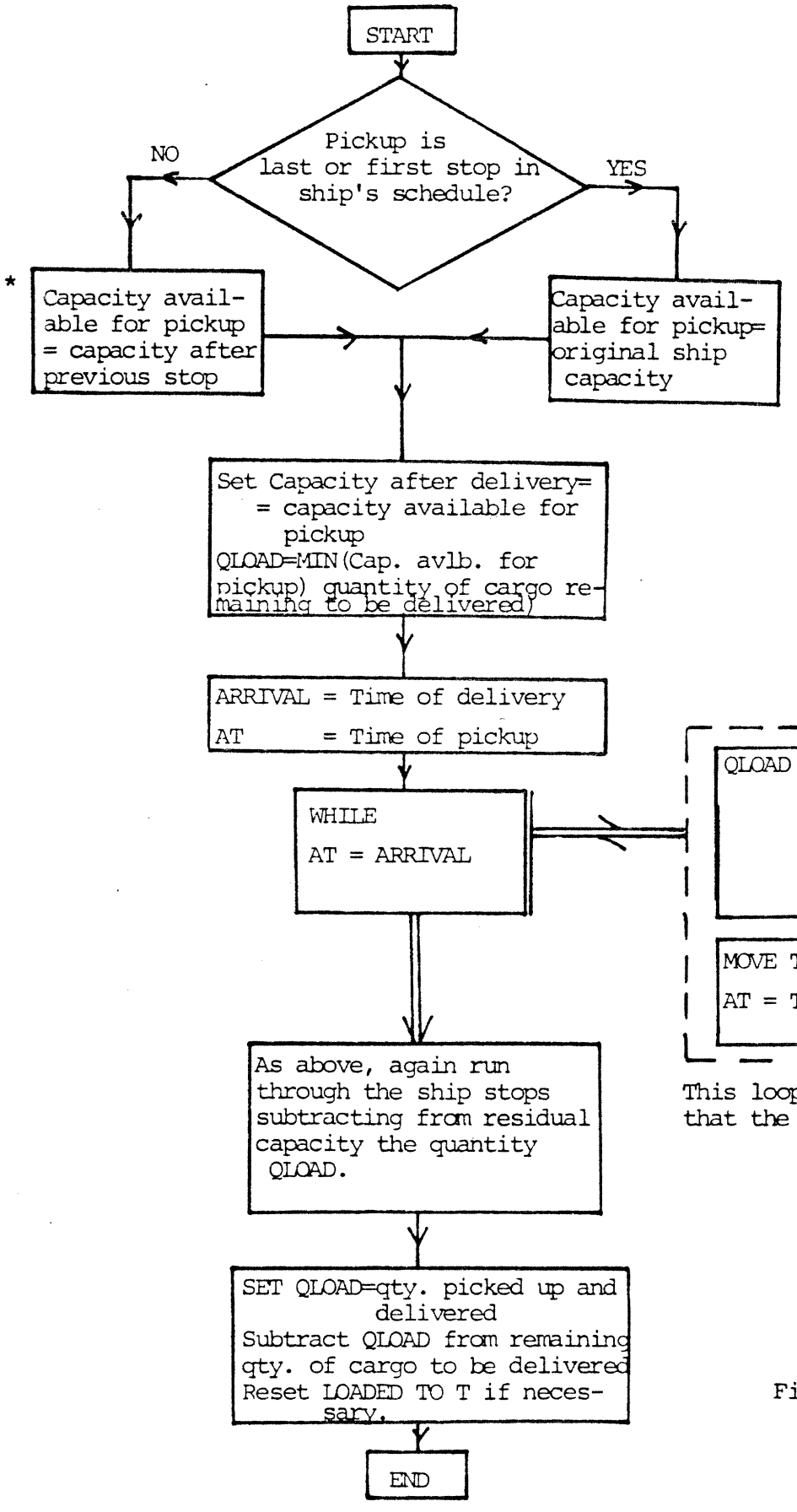

DWN_LOAD
START
Set
CONDIT = F
BANNER = F
CONDIT: Auxiliary boolean variable
WHILE
NOT CONDIT
END
Download next cargo from cargo list:
SEEN: = SEEN +1
IF
EPT ≤ T and NOT LOADED
Cargo is added to current time window only if EPT≤T and the cargo wasn't previously loaded (as a seed)
ELSE
THEN
IF
EPT > T
THEN
CRGC: = CRGC + 1
Add cargo to cargo stack
SEEN:=SEEN-1
CONDIT:=TRUE
We stop downloading as soon as EPT>T or we reach the limit
IF CRGC = LIMIT
THEN CONDIT: = TRUE
IF
SEEN=NCRG
Have we reached end of overall cargo list?
THEN
CONDIT = TRUE
BANNER: = TRUE

Figure B.11

QUPD



* We also set capacity after pickup equal to this quantity

This loop sets QLOAD to be the maximum that the ship can load of the cargo

Figure B.12

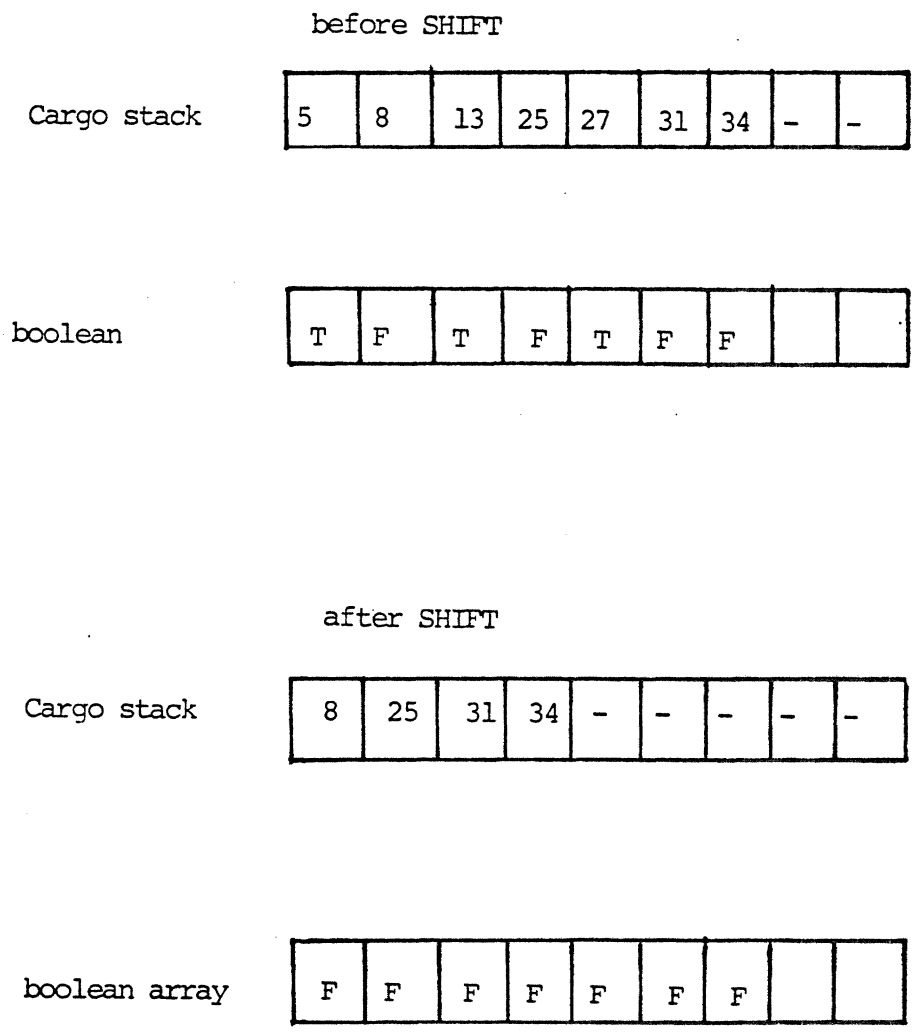

before SHIFT
Cargo stack
5
8
13
25
27
31
34
-
-
boolean
T
F
T
F
T
F
F
after SHIFT
Cargo stack
8
25
31
34
-
-
-
-
-
boolean array
F
F
F
F
F
F
F

Figure B. 13.1

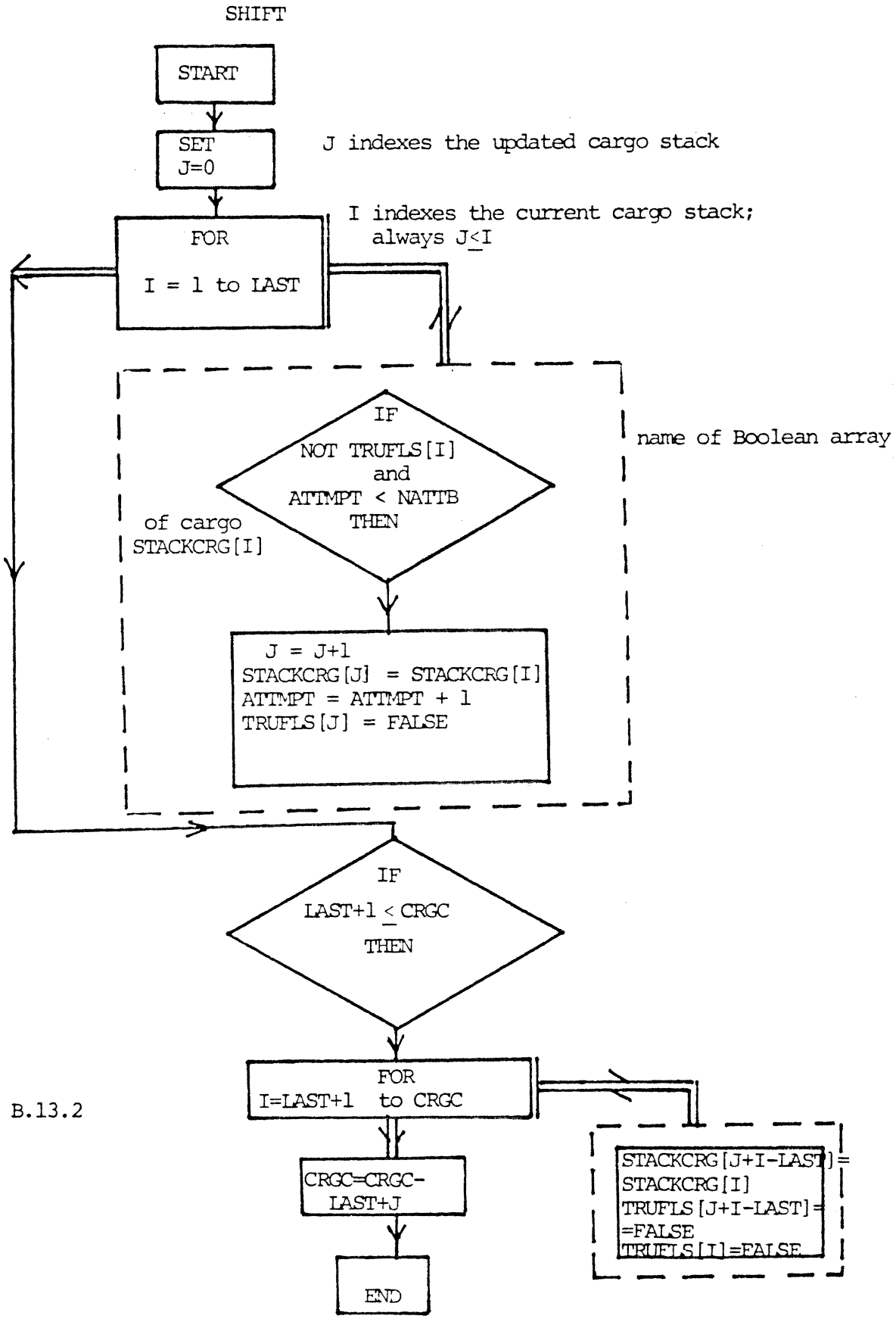
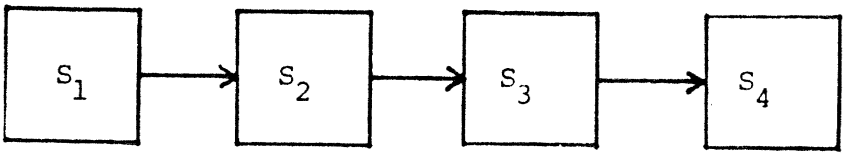
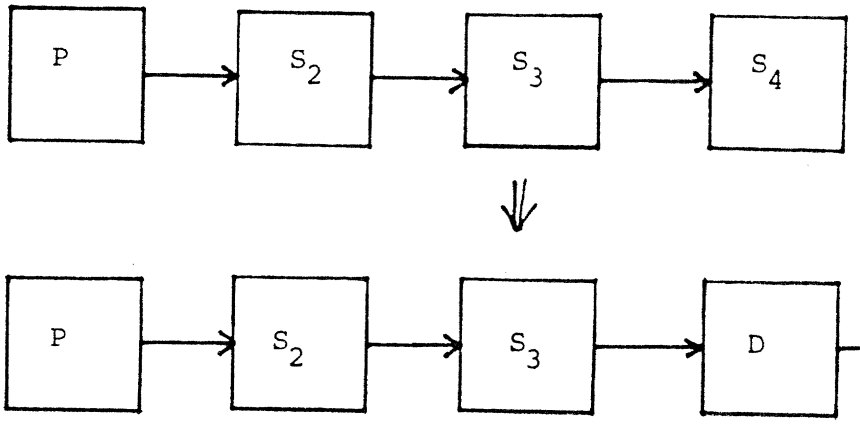

SHIFT
START
SET
J=0
J indexes the updated cargo stack
FOR
I = 1 to LAST
I indexes the current cargo stack;
always J≤I
IF
NOT TRUFLS[I]
and
ATTMPT < NATTB
THEN
of cargo
STACKCRG[I]
name of Boolean array
J = J+1
STACKCRG[J] = STACKCRG[I]
ATTMPT = ATTMPT + 1
TRUFLS[J] = FALSE
IF
LAST+1 ≤ CRGC
THEN
FOR
I=LAST+1 to CRGC
CRGC=CRGC-
LAST+J
END
STACKCRG[J+I-LAST]=
STACKCRG[I]
TRUFLS[J+I-LAST]=
=FALSE
TRUFLS[I]=FALSE

Figure B.13.2



(a)



(b)

Figure B.14.1

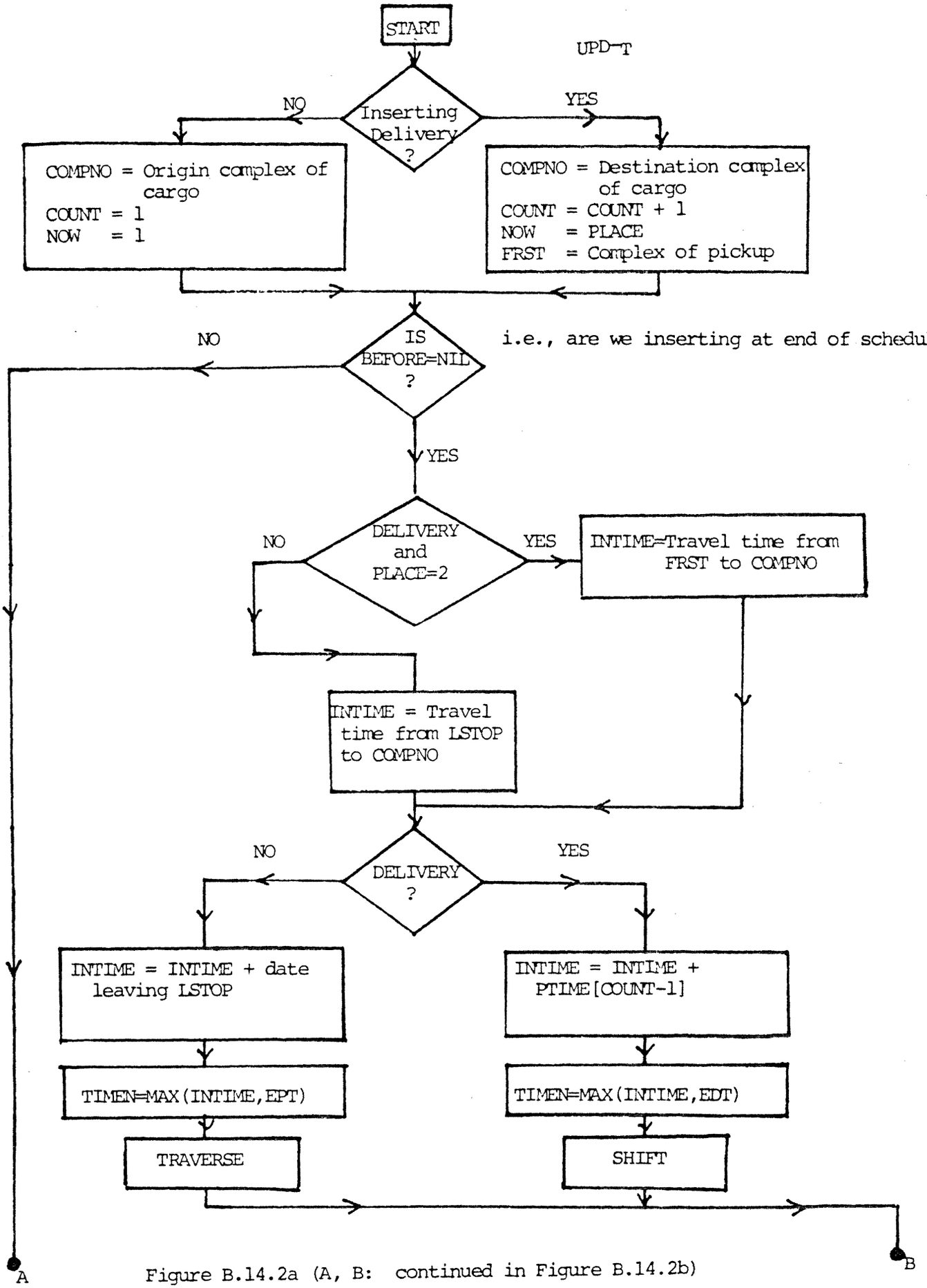


Figure B.14.2a (A, B: continued in Figure B.14.2b)

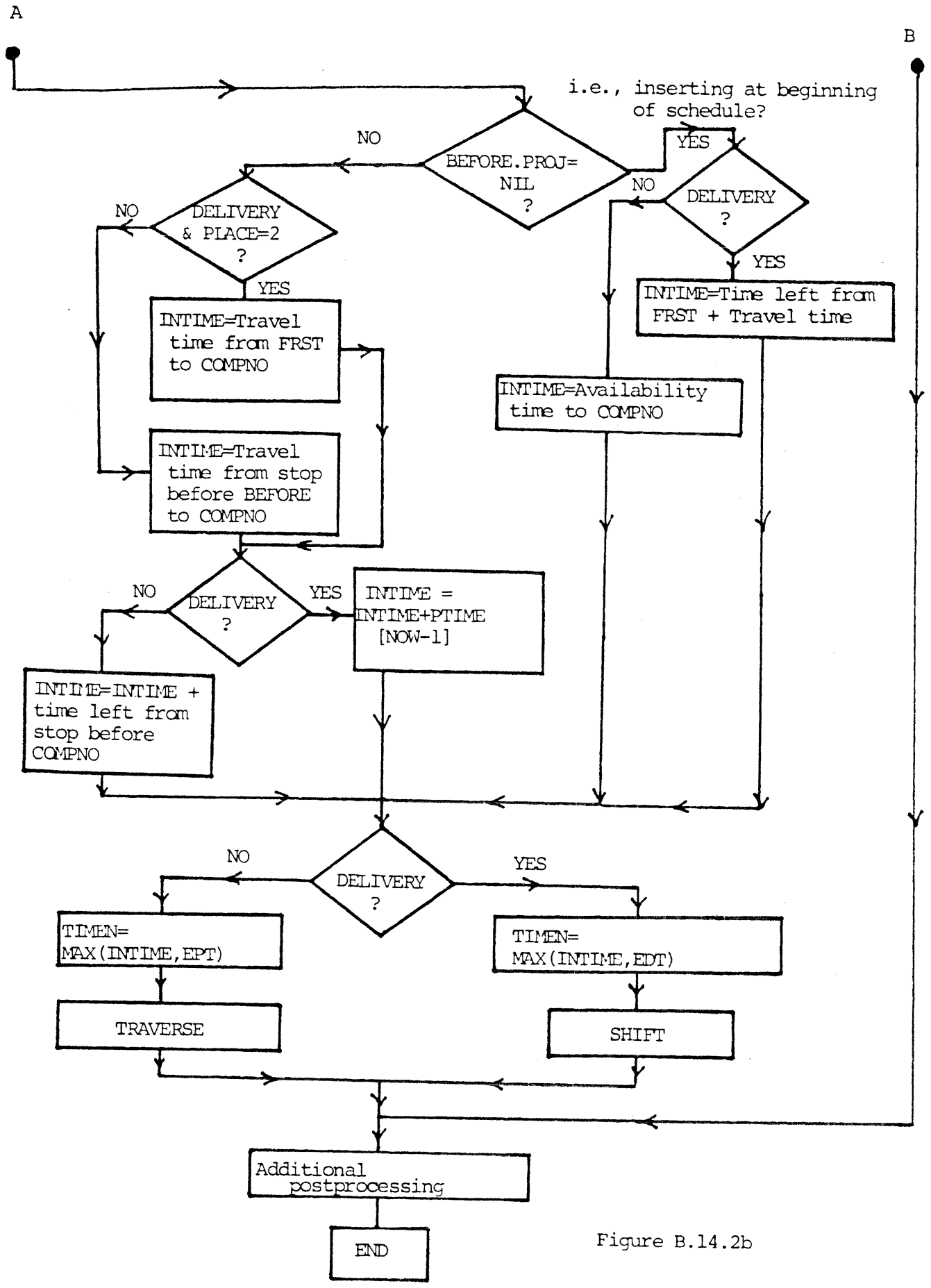


Figure B.14.2b

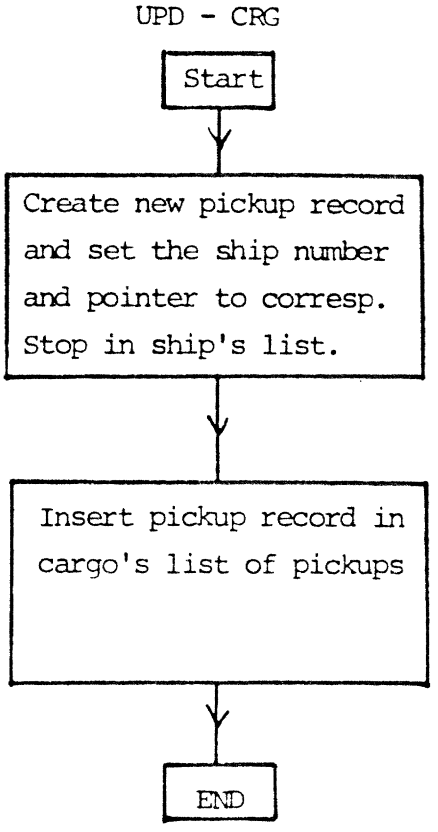

UPD - CRG
Start
Create new pickup record and set the ship number and pointer to corresp. Stop in ship's list.
Insert pickup record in cargo's list of pickups
END

Figure B.15

APPENDIX C

DESCRIPTION OF ARGUMENTS OF PROGRAMS

The following convention is used:

+ = input to program and untouched by program

- = created by program

* = input to program and changed by program)

READIN (VAR I-01:INTEGER; VAR COMPLEX:HUBS;
VAR SUPRCMP:ALLAGGC; VAR NUMSUP,NUMCRG,NUMSHP:INTEGER
VAR TO-MVT:MVTPTR; VAR SHIPS:BOATPT)

-I-01: variable input by user during execution of READIN and later passed to other programs. This variable is used for output control. O I-01 10. The higher I-01 is, the more output (of assignment mechanics, etc.) the user will see.

-COMPLEX: array of complex records

-SUPRCMP: array of supercomplex records

-NUMSUP, NUMCRG, NUMSHP: numbers of supercomplexes, cargoes and ships

-TO-MVT: array of pointers to cargo records.

-SHIPS: array of pointers to ship records.

SEEDS (I-01: INTEGER; VAR STACKSHP: ARRSHP;
VAR COMPLEX: HUBS; VAR SUPRCOMP: ALLAGC; VAR SHPNUZ: INTEGER
VAR TO-MVT: MVTPTR; VAR SHIPS:BOATPT)

+ I-01

-STACKSHP: array created by SEEDS which contains the codes of ships to which cargoes are assigned as seeds.

+ COMPLEX

+ SUPRCOMP

- SHPNUZ: number of ships to which cargoes are assigned

* TO-MVT: records of seed cargoes are modified

* SHIPS: a schedule is started for each ship with a seed cargo

SCHEDULE (I-01:INTEGER; VAR STACKSHP: ARRSHP; VAR COMPLEX: HUBS;
VAR SUPRCOMP: ALLAGGC; VAR SHPNUZ: INTEGER; NUMCRG:INTEGER
VAR TO-MVT: MUTPTR; VAR SHIPS: BOATPT)

+ I-01
+ STACKSHP
+ COMPLEX
+ SUPRCOMP
+ SHPNUZ
+ NUMCRG
* TO-MVT: cargo assignments will be reflected in the cargo data structures
* SHIPS: same as above, for ships.

RLLTIME (I-01:INTEGER; VAR CRGC, SEEN, LIMIT, NUMCRG,T: INTEGER VAR BANNER:
BOOLEAN; VAR STACKCRG: ARRCRG; VAR TRUFLS: BARR1; VAR TO-MVT: MVTPTR)

+ I-01

* CRGC: number of cargoes in cargostack will generally increase

* SEEN: number of cargoes seen by algorithm will generally increase.

+ LIMIT:

+ NUMCRG:

* T : user input.

* BANNER: will be set T if we reach end of cargo array

* STACKCRG

* TRUFLS: this array of boolean variables corresponds to the cargo
stack they are set to F initially, look under POSTOPT).

+ TO-MVT:

```
NETWORK (I-01: INTEGER; VAR STACKCRG:ARRCRG;  VAR STACKSHP: ARRSHP; VAR
     COMPLEX: HUBS;   VAR SUPRCOMP: ALLAGGC; VAR CRGC; SHPNUZ: INTEGER;
     VAR AL,BT: LINKS; VAR TO-MVT:MVTPTR; VAR SHIPS: BOATPT;  VAR FLOWS:
     ALLF; VAR TOTAL: INTEGER)

+ I-01
+ STACKCRG: stack of cargoes
+ STACKSHP: stack of ships
+ COMPLEX
+ SUPRCOMP
+ CRGC, SHPNUZ: # of cargoes and ships in their stacks
+ AL,BT: auxiliary pointers used by ASSNUTIL to compute utilities
+ TO-MVT: array of all cargoes
+ SHIPS: array of all ships
- FLOWS: variable of type ALLF, which itself is an array (1..N)
        of PRSHP records.  Each PRSHP record tells us what was
        assigned to a given ship.  Each record looks like
        SNUM: integer, number of ship (in STACKSHP)
        CNUMS: array (1..4) of integer, number of cargoes assigned
               to ship (also in STACKCRG)
        PICKS: array (1..4) of links of pointers to pickup
               point in ship's schedule, one per assigned cargo
        DELVS: as with PICKS, for delivery points
        COUNT: integer, total number of cargoes assigned to ship
- TOTAL :total number of ships to which cargoes were assigned.
```

PERMASSN (I-01: INTEGER; VAR AL, BT: LINKS; VAR STACKCRG: ARRCGR; VAR
STACKSHP: ARRSHP; VAR TRUFLS: BARR1; VAR COMPLEX: HUBS; VAR SUPRCMP:
ALLAGGC; VAR TO-MVT: MUTPTR; VAR SHIPS: BOATS VAR FLOWS: ALLF; VAR
TOTAL: INTEGER)

+ I-01

+AL, BT: auxiliary pointers utilized by ASSNUTIL

+STACKCRG

+STACKSHP

*TRUFLS: BARR1 (refer to Appendix A. Utilized by
ASSIGN and later in RLLTIME.

+ COMPLEX

+ SUPRCMP

* TO-MVT: cargo records will be modified to show assignments

* SHIPS: ship records will be modified to show assignments

+ FLOWS: raw assignment data from optimization

+ TOTAL: number of ships to whom cargoes have been assigned in optimization

ASSIGN (I-01: INTEGER; VAR TOSHP: LINK5; VAR TOCRG:

LINKS VAR BPCKUP; BDELV: LINK5; VAR COMPLEX: HUBS)

+ I-01

* TOSHP: ship's schedule is updated to include stops

* BPCKUP: pointer to record in ship's schedule, before

which cargo pickup will occur.

*BDELV: as BPCKUP, for delivery

+ COMPLEX

```
UPD-P (I-01: INTEGER; DELIVERY: BOOLEAN; VAR TOSHP: LINKS;
      VAR TOCRG: LINK1; VAR BEFORE, PMVT: LINKS; VAR COMPLEX: HO

+  I-01
* DELIVERY: = T if stop to be inserted is a delivery
* TOSHP: ship's schedule is modified
* TOCRG: cargo's pickup list modified
+ BEFORE: pointer to stop immediately following insertion
-PMVT: pointer to inserted stop (manufactured by UPD-P)
+ COMPLEX
```

DWN-LOAD (I-01: INTEGER; VAR STACKCRG: ARRCRG; VAR CRGC, SEEN: INTEGER;
LIMIT, NCRG, T: INTEGER; VAR TRUFLS: BARR1; VAR BANNER: BOOLEAN; VAR
TO-MVT: MVTPTR)

+ I-01

* STACKCRG: new cargoes are generally down loaded onto stack.

* CRGC: cargo count = number of cargoes in stack.

* SEEN: number of cargoes that have been scanned in overall list.

+ LIMIT: upper bound on number of cargoes in stack (= time window).

+ NCRG: total number of cargoes in overall list.

+ T : upper bound on EPT's in time window being built.

* TRUFLS: this array utilized in SHIFT, page

* BANNER: becomes TRUE if we reach end of overall cargo list.

+ TO-MVT

QUPD (I-01: INTEGER; VAR PCKUP, DELU: LINKS; VAR TOSHP: LINKS; VAR TOCRG: LINK1)

+ I-01

* PCKUP: pointer to pickup record. Record changed to reflect quantity picked up.

DELV: pointer to delivery record. Record changed to reflect quantity delivered.

*TOSHP: pointer to ship. Ship's schedule changed (capacities reduced).

* TOCRG: pointer to cargo.

SHIFT (I-01: INTEGER; VAR STACKCRG: ARRCRG; VAR TRUFLS:
BARR1; VAR CRGC, LAST: INTEGER, VAR TO-MVT: MVTPTR)

+ I-01

* STACKCRG: ATTMPT will be increased by 1 for some cargoes

* TRUFLS: Remaining cargoes will be made into F's

* CRGC: Updated

+ LAST

* TO-MVT

```
UPD-T (I-01:INTEGER; DELIVERY: BOOLEAN; VAR COUNT:
      INTEGER; VAR AL, BET: LINK3; VAR NUTIME, DTIME,
      NUSLK, PSLK, OUTSLK: ARSHP; VAR TOREC, PTOREC: PARRZ;
      VAR TO-MVT: MVTPTR; VAR COMPLEX: HUBS)

+I-01
+DELIVERY: BOOLEAN, = T if inserting delivery
* COUNT: length of attempted schedule as measured from pickup
* AL,BET: auxiliary pointers, used to store pickup and delivery
* NUTIME,...,OUTSLK: arrays to hold time and slack data
*TOREC, PTOREC: arrays to hold stop pointers for schedule
+ TO-MVT
+ COMPLEX
```