

A SOLUTION PROCEDURE FOR THE  
STRATEGIC TRANSPORTATION PROBLEM

A THESIS

Presented to  
The Faculty of the Division of Graduate  
Studies and Research

By  
Peter Dean Keith

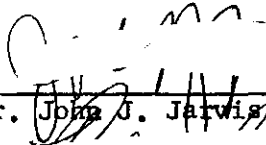
In Partial Fulfillment  
of the Requirements for the Degree  
Master of Science in Operations Research


Georgia Institute of Technology

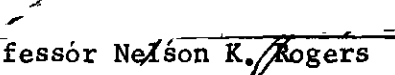
November, 1973

A SOLUTION PROCEDURE FOR THE  
STRATEGIC TRANSPORTATION PROBLEM

Approved:

  
\_\_\_\_\_  
Dr. John J. Jarvis, Chairman

  
\_\_\_\_\_  
Dr. R. G. Parker

  
\_\_\_\_\_  
Professor Nelson K. Rogers

Date Approved by Chairman: 11/28/73

## ACKNOWLEDGEMENTS

My thanks go to many for their guidance and encouragement during this research and my stay at Georgia Institute of Technology, especially to Dr. John J. Jarvis, who served as both academic advisor and chairman of my thesis committee, and to Dr. R. G. Parker and Professor Nelson K. Rogers, who served as readers on the thesis committee. Another special thank you must go to Mrs. Nancy Price for her help during the typing of this thesis. And finally, to all my friends in the Industrial and Systems Engineering department, a quote from John Lennon and Paul McCartney, "I get by with a little help from my friends".

## TABLE OF CONTENTS

	Page
ACKNOWLEDGEMENTS . . . . .	ii
LIST OF FIGURES . . . . .	v
SUMMARY . . . . .	vi
CHAPTER	
I. INTRODUCTION . . . . .	1
Statement of Problem	
Example	
Proposed Solution Method	
Bases-to-Ports Transportation Problem	
Ships-to-Ports Scheduling Problem	
Branch and Bound Coupling Procedure	
Assumptions and Definitions	
Objectives	
II. LEAST TIME TRANSPORTATION PROBLEM . . . . .	13
A General Algorithm to Solve the Least Time Transportation Problem	
Example	
III. NETWORK CONSTRUCTION FOR THE SHIPS-TO-PORTS PROBLEM. .	22
Ship Scheduling Network Construction Algorithm	
Algorithm for the Application of Lower and Upper Capacities, and Lengths to Arcs of the Ship Scheduling Network	
Formulation of the Ships-to-Ports Subproblem	
IV. DISCUSSION OF HINKLE'S WORK . . . . .	35
A General Min-Max Path Flow Algorithm	
Determining Maximal Flow through Capacitated Networks	
Extensions of Hinkle's Algorithm for the Ships-to-Ports Subproblem of Chapter III	

## TABLE OF CONTENTS (Continued)

	Page
V. CONSTRAINT SWITCHING . . . . .	54
The Subopt Procedure Applied to the Maximal Flow Problem	
Standard Linear Programming Approach to the Maximal Flow Problem	
Theory of Constraint Switching in the Problem of Maximal Flow through a Network with Lower and Upper Arc Capacities	
Constraint Switching Algorithm	
Convergence	
VI. THE CONSTRAINED SHORTEST PATH ALGORITHM FOR ACYCLIC NETWORKS . . . . .	78
The Constrained Shortest Path Algorithm	
Example	
VII. EXAMPLE . . . . .	86
VIII. PROCEDURES FOR COUPLING THE TWO SUBPROBLEMS TO TOGETHER . . . . .	93
Branch and Bound	
Utilizing Linear Programming Information in the Branch and Bound Procedure	
Total Travel Time Utilized in a Hueristic Solution Procedure	
IX. CONCLUSIONS AND RECOMMENDATIONS . . . . .	114
A Summary of Research Results	
Other Applications	
Extensions and Areas of Further Research	
APPENDIX A . . . . .	128
APPENDIX B . . . . .	155
BIBLIOGRAPHY . . . . .	167

## LIST OF FIGURES

Figure	Page
1. The General Strategic Transportation Problem . . . .	3
2. A Specific Example of the Strategic Transportation Problem . . . . .	10
3. A Network With Only Upper Arc Capacities and Arc Costs . . . . .	36
4. The Ship Scheduling Network for the Example . . . .	89
5. The Ship Scheduling Network for the Example, Renumbered . . . . .	90
6. A Portion of a Tree Created by a Branch and Bound Algorithm . . . . .	96

## SUMMARY

This thesis summarizes the research results on a class of closure networks. A solution procedure, for a strategic transportation problem is presented, which includes a decomposition of the total problem into two subproblems and algorithms to solve each of these subproblems. One algorithm presented handles lower bounds on arc flows in the context of a maximum flow network problem. This algorithm employs a constraint switching operation coupled with the standard methodology of decomposition in linear programming, to allow only  $m$  (number of arcs in the network) constraints in the constraint set and thus a basis of size  $m$  by  $m$ . A least time transportation algorithm is also used in the solution procedure, along with a branch and bound scheme to couple the two subproblems. Some basic linear programming relationships are employed to change the requirement vector of the transportation problem, thus allowing an iterative solution procedure to the strategic transportation problem. A heuristic algorithm is also examined as a possible coupling procedure between the two subproblems.

An efficient method of solving a constrained shortest path problem is also presented. This procedure handles negative costs on the arcs of an acyclic network. It is a modification of the labelling technique proposed by Hinkle and it allows a one pass solution for column generation.

## CHAPTER I

### INTRODUCTION

The objective of this thesis is to report the results of research on a class of network problems, known as minimal closure problems.

Stated briefly, the problem is to determine the flow through a capacitated network such that arrival time of the last unit of flow from source to the sink is minimized.

Before discussing an algorithmic procedure to solve this type of problem, an example of a minimal closure problem is presented below. This example will be used throughout this thesis as a medium, through which the solution methodologies will be presented.

#### Example

A well known military problem, called a strategic transportation problem or resupply problem, falls into the class of minimal closure problems. In a strategic transportation problem there are commodities or goods which are stored or based at inland supply depots. There are also a number of coastal ports to which these goods can be moved from the supply depots, over associated transportation links. Once the goods have been transported to the ports, they are to be loaded on ships and moved to a pre-determined resupply point or objective area. The ships, however, are located at sea and must travel to a port in order to take on cargo before they can move onward to the objective area. Once the supplies reach the objective area, they are either stored temporarily



or used immediately to resupply a military force. In either case, however, the strategic transportation problem ends once these supplies have reached the objective area. Since the strategic transportation problem is concerned with the movement of supplies and ships through time, it defines an initial time reference point (i.e. the model begins at a zero time point). So a strategic transportation problem may be stated as: given that there are known quantities of supplies at the inland bases we wish to assign the supplies at each port, such that the ships may be scheduled to move to the ports, take on cargo, and travel to the objective area in such a manner that the last ship will arrive at the objective area in a minimum amount of time (i.e. minimal closure with respect to the initial time reference point). A graph of the strategic transportation problem, showing the relative placement of the bases, ports, ships, and the objective area; along with all the associated transportation links is presented in Figure 1.

#### Proposed Solution Method

The strategic transportation problem seems to contain a very intuitive separation scheme. The supplies must first be transported from the inland depots to the ports, and then loaded onto ships for transportation to the objective area. Since supply storage may occur at the ports, a natural decomposition of the problem can occur at the ports, that is divide the total problem into two subproblems. The first subproblem is that of transporting the supplies from the inland bases to the ports and the second subproblem is the scheduling of the ships to the ports. Included in the second subproblem is the scheduling of

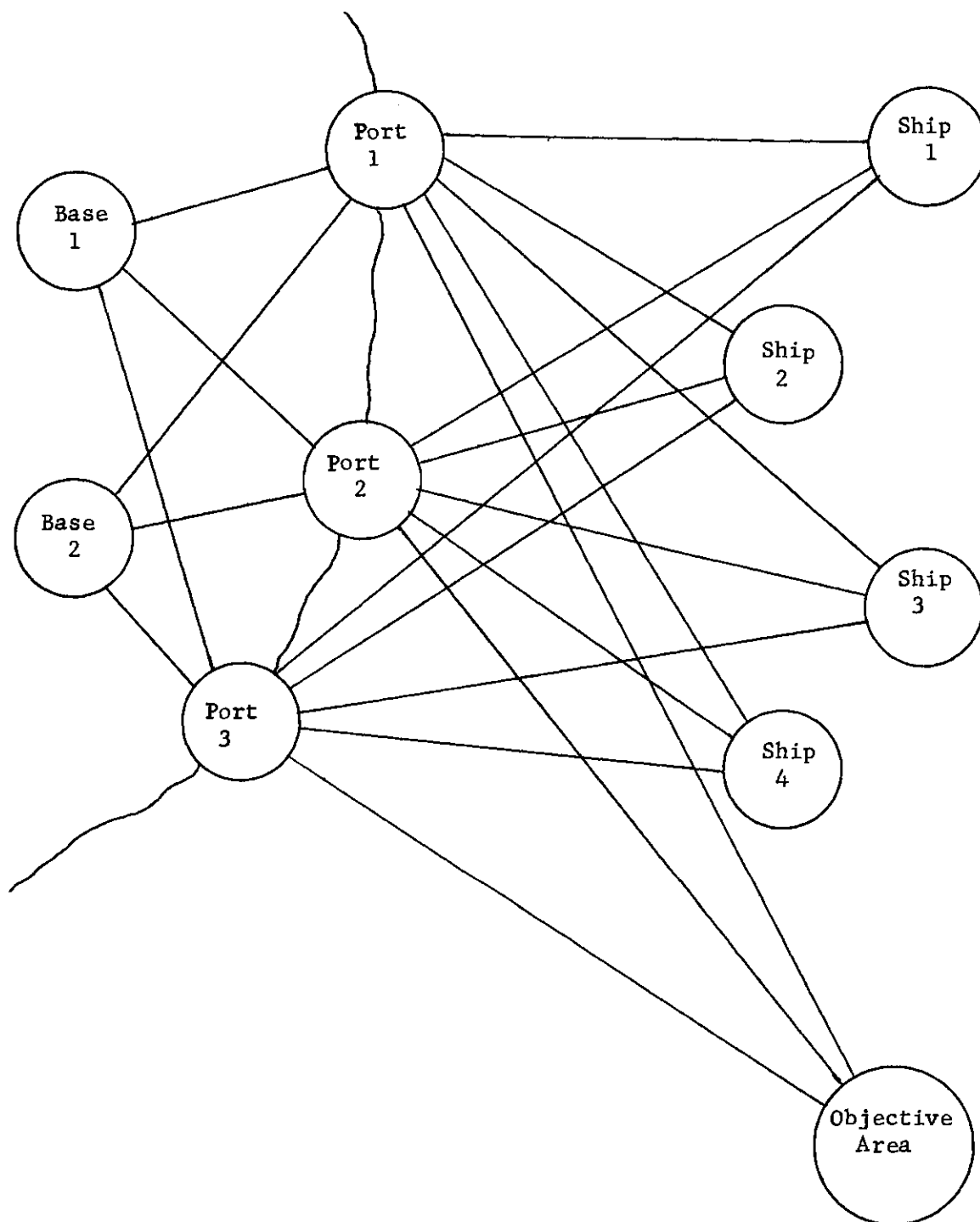


Figure 1. Strategic Transportation Problem.

the ships to the objective area. The first subproblem could be solved by a standard transportation model. However, the ship scheduling is dependent upon the allocation of supplies to the ports and the times at which these supplies arrive. A ship may not leave a port for the objective area until it has been loaded with supplies. Thus for a specified allocation of supplies to the ports, the supplies must be transported in a minimum amount of time. Therefore, the first subproblem must take into consideration this constraint of minimum transportation time of the supplies in addition to satisfying the demand requirements at each port. After determining the quantities and arrival times of the supplies at each port, we can then solve the second subproblem. If ships are considered as units of flow and the supplies at each port are considered as capacities on arcs, the ship scheduling subproblem becomes a maximal flow network problem through a capacitated network. The second subproblem must also consider minimization of transportation time since the closure time for the total strategic transportation problem is defined as the arrival of the last unit of supplies at the objective area. Thus the second subproblem must determine the actual schedule of the ships to each port. Since the first subproblem must explicitly transport supplies and the second subproblem is implicitly transporting supplies, the connecting link between the subproblems is the allocation of supply requirements to each port (i.e., what quantities are required at each port).

There are a number of other transportation problems, which can be classified as minimal closure problems, that exhibit the same decomposition capability as the strategic transportation problem. The perishable

goods industry is another example of this problem. Suppose a producer of canned foods has a number of collection centers located throughout a crop producing area, and one canning factory located near a major urban area where the finished products will be sold. At harvest time the producer purchases the crops from a number of farmers in the growing area and now the crops must be moved through the collections centers and onto the factory before spoilage destroys the crops. The producer would like to transport the crops to the collection centers, have them loaded onto carriers (i.e. trucks or railroad cars for instance), and have the carriers arrive at the canning factory such that the last of the crops to arrive, will not spoil.

#### Bases-to-Ports Transportation Problem

Let the first subproblem of the strategic transportation problem be called the Bases-to-Ports transportation problem. In general terms this subproblem is the transporting of the supplies to the ports so that they can be available for loading when the ships arrive. The method utilized in this thesis to solve the Bases-to-Ports subproblem is an application of the Least-Time Transportation Algorithm. The approach taken by the least-time transportation-type problem, is to determine the minimal time set of travel links between the sources and the destinations, such that all of the requirements for supplies at the ports are met. At each iteration, cells with the longest time carrying flow and cells with longer times are penalized. These cells with longer times are effectively removed from further consideration by assigning them arbitrarily large times. The flow on the cells with the largest

times of the cells carrying flow, is successively reduced until no more decreases can take place. By reducing the flow on the larger time cells, the least time transportation algorithm determines the minimal time set of cells that will allow the transportation of the supplies from the bases to satisfy the port requirements for supplies. In addition, this reduction process of the least time transportation algorithm assures that all of the supplies arrive at the ports in a minimal amount of time (i.e. minimal closure is assured for the Bases-to-Ports transportation problem).

#### Ships-to-Ports Scheduling Problem

Let the second subproblem of the strategic transportation problem be called the Ships-to-Ports scheduling problem. If all of the ships are located at sea when the strategic transportation problem begins (i.e. at time zero) this subproblem is concerned with getting the ships to the ports as soon as is feasible, loading the supplies and dispatching the loaded ships to the destination area. An additional objective of the Ships-to-Ports problem is that the arrival time of the last ship at the objective area is minimized. The actual scheduling problem could be formulated as a maximal flow problem through a capacitated network. However, this additional objective of minimal closure (i.e. arrival time of the last ship is minimized) requires the minimization of the maximal length path, carrying flow through the network. Hinkle (1) has described this problem as a min-max network flow problem. His algorithm requires the solution of a series of linear programming problems. His procedure works in a similar manner to the least-time transportation

algorithm in that from each linear programming problem to the next, the longest path carrying flow and all paths at least as long are penalized. These transitions effectively take place through the determination of a path, of shorter length, to enter the basis that will reduce the flow on the longest path in the basis. A constrained shortest path algorithm is utilized to develop such a "reducing" path. The application of Hinkle's algorithm to the Ships-to-Ports scheduling problem assures the arrival time of the last ship at the objective area is minimized (i.e. minimal closure for this subproblem).

#### Branch and Bound Coupling Procedure

The decomposition of the strategic transportation problem, occurs at the ports, in that the first subproblem is considered with transporting the supplies from the bases to ports, and the second subproblem is considered with transporting these supplies from the ports to the objective area. Hence the requirements for supplies at the ports become the connecting link between the subproblems. It may be the case that varying the destination and/or transportation times of certain supplies from the bases might lead to a more favorable scheduling of ships through the ports in such a way the overall closure time (the time the last ship arrives at the objective area) is reduced. This situation would require the changing of the requirements for supplies at the ports and, in turn, through the new solution to the least time transportation problem the new arrival times of the supplies would generate a schedule of ships.

### Assumptions and Definitions

The supplies at the inland bases are assumed to be packaged in standardized containers, which may contain a single type of supply or a mixture. However, when the resupply becomes necessary these containers will be brought out of storage and transported, not the individual supplies. The use of the term, supplies, will imply these standardized containers of supplies. A common unit of supplies must be defined, for use in both of the subproblems of the strategic transportation problem. The common unit will be a ship load of supplies, that is the number of containers that a ship can be loaded with (i.e. if a base has three units of supplies stored, it would have enough containers of supplies to load three ships). Implicit in the definition of the common unit to be transported, is the assumption that each ship is of the same size and has the same load carrying capacity. It is also assumed that each ship travels at the same speed. Therefore all of the ships are part of a standardized fleet, in that the size, speed, and load carrying capacity of each ship is identical. We further assume that the ships are randomly located at sea, that is to say, all of the ships are not anchored near one port, but they are travelling at sea, not as one fleet, but either as individual ships or as small groups. However, the precise location of each ship is known, and it is denoted by the time required to travel from its location to any of the coastal ports. If supplies are available at a particular port when a ship arrives, the containerized cargo is immediately loaded and the ship can then sail for the objective area. Hence at each port there does not exist a berth capacity constraint, and there is assumed to be adequate storage for any supplies

that must wait for a ship to arrive.

The reader may be curious, at this point, as to why all of these simplifying assumptions about the strategic transportation problem were made. The military<sup>+</sup> has been studying the problem of resupply and has proposed that in the 1980-1990 time frame of reference that (1) containerized shipments of supply will be generated in ship load quantities from inland container storage bases; (2) the operation of ports will be, simply to transfer these containerized shipments from one mode of transportation to another; and (3) the shipments of supplies will be transported by standard size container vessels. Therefore, the strategic transportation problem discussed at the beginning of this chapter, becomes the problem the military is concerned with for the future.

A graphical example of the strategic transportation problem is now presented. There are five units of supplies stored at the inland bases, one unit at base one and four units at base 2. The port requirements are two units at port one and three units at port two. The travel times from the bases to the ports, from the ship locations to the ports, and from the ports to the objective area are indicated on Figure 2. Two units of supplies will arrive in port one at time 3, one unit will arrive in port two at time 2, and two units will arrive in port two at time 4. To obtain the minimal closure for the total strategic transportation problem, the ships must be scheduled as follows: ship one to port two, ship two to port two, ship three to port one, ship four

---

<sup>+</sup>Preliminary discussions concerning the future resupply systems have been conducted by the military. However, the specific results of these discussions are not presently available to the public.



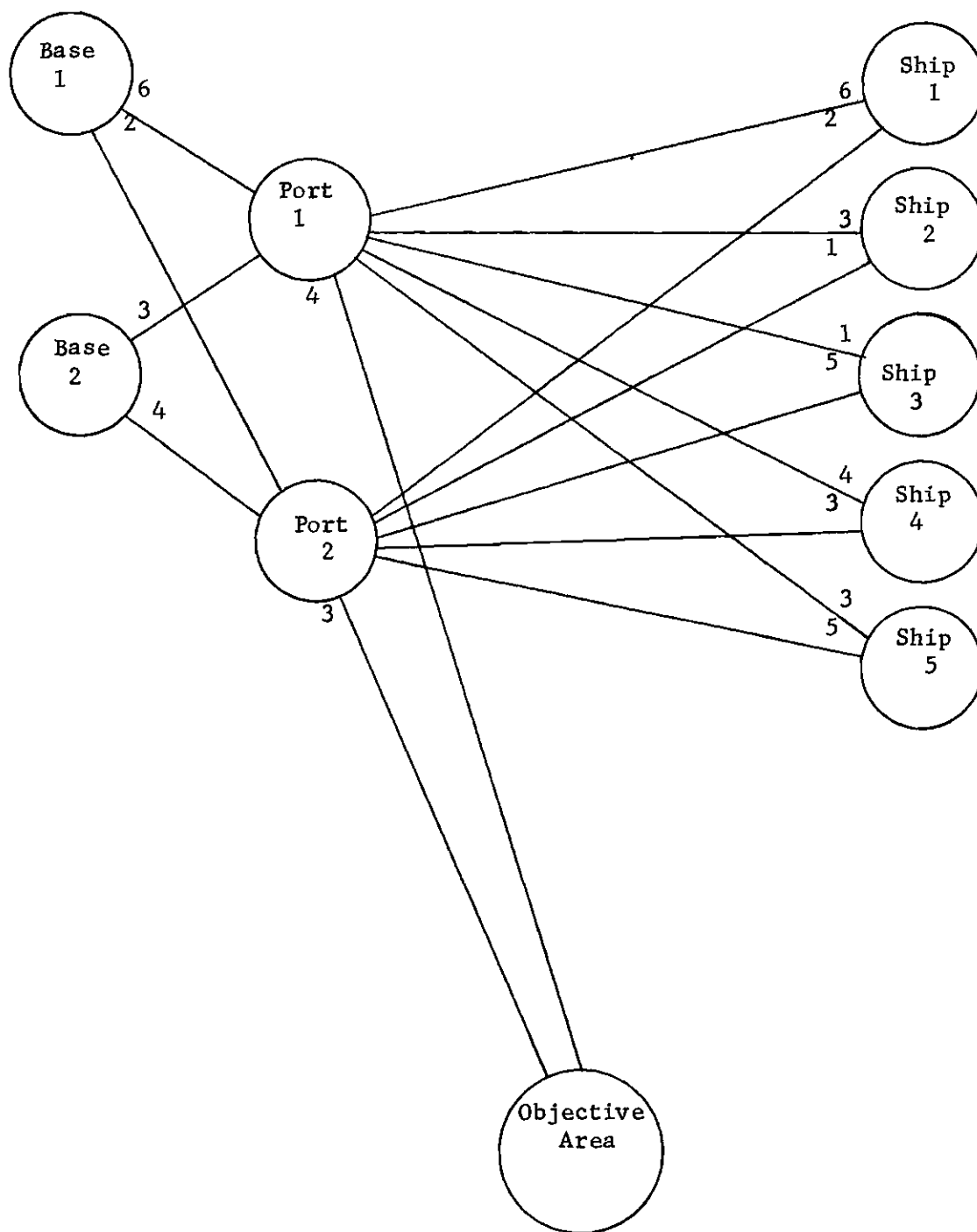


Figure 2. A Specific Example of the Strategic Transportation Problem.

to port two and ship five to port one. The closure (i.e. the arrival time of the last unit of supplies at the objective arc) is seven time units, and occurs since ships two through five will arrive at the objective area at time seven.

### Objectives

The objective of this research was to investigate and characterize a strategic transportation problem as one type of minimal closure network problems, and to develop a computationally feasible algorithm to solve this problem. This problem has been formulated, in terms of the future military resupply problem with the hope that intelligent decisions can be made with respect to the economic feasibility of this type of supply system, in light of the solution procedure presented in this thesis.

Specifically, a decomposition technique is investigated to generate the two subproblems of the strategic transportation problem. The Bases-to-Ports subproblem is formulated as a modified standard transportation problem and a least time transportation algorithm is applied for the solution of it. The Ships-to-Ports subproblem can be formulated as a maximal flow problem through a capacitated network. Hinkle's min-max path flow algorithm could be used in the solution of this subproblem, if the algorithm allowed lower capacities in the network. His algorithm is considered and an extension is also investigated to handle lower capacities in the maximal flow network problem. His "reducing" path generation algorithm is examined in the context of the lower capacities in the network to determine if any modifications

are necessary and if the algorithm can be made computationally more efficient for the application of the strategic transportation problem. Finally, two schemes are investigated as possible coupling procedures between the two subproblems of the strategic transportation problem. One of these schemes is a branch and bound algorithm and the other is a heuristic algorithm.

## CHAPTER II

### LEAST TIME TRANSPORTATION PROBLEM

In Chapter I it was stated that the strategic transportation problem could be decomposed into two subproblems. One of these subproblems is the Bases-to-Ports transportation problem. A general transportation problem is concerned with moving some commodity from one group of locations, which are known as sources, to another group of locations, which are known as destinations. Different quantities of the particular commodity are stored at each source, however, these quantities are known for each source. There is also an amount of the commodity that is required at each of the destinations, and again each of these required quantities are known. With each source there is an associated set of travel links which connect the source to any one of or all of the destinations. One of these travel links might be an airline route, a highway, or a rail line which connect one of the sources to one of the destinations, and there is a cost associated with moving a unit of supplies along each of these links. The problem now becomes one of transporting all of the commodity that is stored at the sources across the travel links to the destinations, so that required amounts at each destination is fulfilled. However, there would exist a number of different ways to meet the commodity requirement at each destination by transporting the commodity across different travel links. If the minimal cost set of travel links between the sources

and the destinations could be determined, and all the commodity requirements were satisfied, then the optimal, minimal cost, solution to this transportation problem would have been found.

However, the objective of the Bases-to-Ports transportation is not to determine the minimal cost set of travel links. It is to determine the set of travel links such that the supplies from the sources reach the destinations in such a way that the arrival time of the last unit of supplies is minimized. If the cost associated with transporting a unit across the travel link is interpreted as the time required to transport a unit across the link, then the standard transportation problem can be formulated as:

$$\begin{aligned}
 (1) \quad & \text{Min } z = \sum_{i=1}^m \sum_{j=1}^n c_{ij} x_{ij} \\
 & \text{st} \\
 & \sum_{j=1}^n x_{ij} = a_i \quad i = 1, 2, \dots, m \\
 & \sum_{i=1}^m x_{ij} = b_j \quad j = 1, 2, \dots, n \\
 & x_{ij} \geq 0 \quad \forall i, j
 \end{aligned}$$

where  $a_i$  = quantity of the commodity stored at source  $i$   
 $b_j$  = quantity of the commodity stored at destination  $j$   
 $c_{ij}$  = cost (or time) of transporting a unit of the commodity  
 from source  $i$  to destination  $j$  over travel link  $(i, j)$   
 $x_{ij}$  = the variables, the quantity of the commodity transported  
 from source  $i$  to destination  $j$  over travel link  $(i, j)$

This transportation problem is known as the time-minimizing transportation problem, because of the special interpretation of the  $c_{ij}$ 's. There are algorithms available in Dantzig (2) and Taha (3) to solve this problem. However, if the transportation problem must determine the minimal arrival time set of travel links (i.e. the set of links such that arrival time of the last unit will be minimized) the formulation is slightly different from (1) and is given below:

$$\begin{aligned}
 (2) \quad & \text{Min} \left\{ \text{Max} \left\{ \delta_{ij} c_{ij} \right\} \right\} \\
 & \forall i, j \\
 & \text{s. t.} \quad \sum_{i=1}^m x_{ij} = a_i \quad i = 1, 2, \dots, n \\
 & \quad \quad \sum_{j=1}^n x_{ij} = b_j \quad j = 1, 2, \dots, m \\
 & \quad \quad x_{ij} \geq 0 \quad \forall i, j
 \end{aligned}$$

where  $\delta_{ij} = \begin{cases} 1, & \text{if } x_{ij} > 0 \text{ (i.e. 1 if travel link (i,j) is used)} \\ 0, & \text{otherwise.} \end{cases}$

This problem is known as the least-time (or min-max) transportation problem.

The reader might think that for a given set of commodity availabilities at the sources and commodity requirements at the destinations, that both problems (1) and (2) will generate the same optimal solution. Let us examine the assignment problem in light of the two formulations given above. The assignment problem is a simplified transportation problem, in that all of the source availabilities ( $a_i$ 's)

and the destination requirements ( $b_j$ 's) are one. If there are three sources and three destinations in the assignment problem, let the time required to transverse the travel links  $(i,j)$  be given in the following matrix,

DESTINATION

		1	2	3
S				
O	1	12	11	11
U				
R	2	11	1	11
C				
E	3	11	11	1

$[c_{ij}] =$

The optimal solution using the time-minimizing transportation model (1) would transport one unit along each of the following links, (1,1), (2,2) and (3,3). This solution is shown below in tableau form.

DESTINATION

		1	2	3
S				
O	1	1	0	0
U				
R	2	0	1	0
C				
E	3	0	0	1

;  $x_{ij}$  matrix.

The objective function  $\sum_{i=1}^m \sum_{j=1}^n c_{ij} x_{ij}$  would have a value of 14, but the unit transported over link (1,1) would arrive at the destination one at time 12. Now if this assignment problem was solved using the least-time transportation model, the optimal solution, in tableau form, would be

		DESTINATION			
		1	2	3	
S					
O	1	0	1	0	; $x_{ij}$ matrix.
U					
R	2	1	0	0	
C					
E	3	0	0	1	

It can be noted that even though this optimal solution has reduced the arrival time of the last unit of the commodity from 12 to 11 ( $\max_{ij} \{c_{ij}\} = 11$ ) the total time required to transport all of the commodity,  $\sum_{i=1}^m \sum_{j=1}^n c_{ij} x_{ij}$ , has increased from 14 to 23. So there exists an important trade-off between the arrival time of the last unit and the total transportation time, that is, in order to obtain the set of travel links which minimizes the arrival of the last unit a penalty in the sense of total time (or cost) must be paid.

There does not seem to exist any connection between the time-minimizing transportation problem and the least-time transportation problem, but the time-minimizing problem can be used as part of the solution procedure of the least-time problem. Since the least-time model determines the set of travel links such that arrival time of the last unit is minimized, a general solution procedure could consist of selecting the minimal cost (or time) set of links to transport the commodity across, penalizing the link with the longest time over which the commodity is transported and any links with longer time and then again selecting the minimal cost set of links. This type of selection and penalization scheme would be continued until the set of links was determined, that did in fact minimize the arrival time of the last unit



at the destinations. If the selection step guaranteed the optimal, minimal cost, set of travel links then the solution procedure for the least-time transportation problem would have the minimum number of selection and penalization steps. Therefore the time-minimizing transportation problem could be used for the selection of the new set of travel links at iteration of the least-time solution procedure. The steps of this algorithm are specifically outlined below:

An Algorithm to Solve the Least Time Transportation Problem

- STEP 1: Start with the initial  $c_{ij}$  values
- STEP 2: Solve a time minimizing transportation problem with the current  $c_{ij}$  values
- STEP 3: Determine if the optimal solution has changed from the previous solution. If YES, penalize the longest time link, over which the commodity is transported and any links with a longer time, with an arbitrarily large time, and return to Step two. In NO, Stop.

This algorithm will reduce the flow of the commodity off the links with the longest transportation time at each iteration until either no further reduction is possible or new set of links with a smaller longest time is determined. If a new set of travel links is found, the algorithm will reduce the flow on the longest time link of this set. Since the algorithm reduces the flow on the longest time link at each iteration, upon termination the set of travel links will have been determined such that the arrival time of the last unit of the commodity at the destinations will be minimal. Another algorithm for solving the least-time transportation problem is presented in Taha (3).

Example

The algorithm described above will now be applied to an example.

Let:

$$a = \begin{bmatrix} 2 \\ 4 \\ 4 \end{bmatrix} \quad b = \begin{bmatrix} 3 \\ 6 \\ 1 \end{bmatrix} \quad c = [c_{ij}] = \begin{array}{c} \text{S} \\ \text{O} \\ \text{U} \\ \text{R} \\ \text{C} \\ \text{E} \end{array} \begin{array}{c} \text{DESTINATION} \\ \\ \\ \\ \\ \\ \end{array} \begin{bmatrix} 3 & 3 & 6 \\ 6 & 4 & 1 \\ 4 & 2 & 2 \end{bmatrix}$$

The optimal solution to time-minimizing transportation problem using the initial  $c_{ij}$ 's is given in tableau form

		DESTINATION			
		1	2	3	
S O U R C E	1	2	0	0	; $x_{ij}$ matrix
	2	1	2	1	
	3	0	4	0	

The longest time, link carrying the commodity is (2,1) and this link along with link (1,3) will be penalized by using a  $C_{ij}$  of 100. The optimal solution to the time-minimizing problem using the new  $c_{ij}$ 's is:

		DESTINATION			
		1	2	3	
S O U R C E	1	3	3	100	$c_{ij}$ matrix
	2	100	4	1	
	3	4	2	2	

		DESTINATION			
		1	2	3	
S O U R C E	1	2	0	0	$x_{ij}$ matrix
	2	0	3	1	
	3	1	3	0	

Now links (2,2) and (3,1) have the longest times of the links carrying flow, so they also will be penalized. The solution to the time-minimiz-

ing transportation problem using the new  $c_{ij}$ 's does not change from the solution given above. Therefore the optimal solution to the least-time transportation problem is also the above solution. From this tableau and the original C matrix it can be seen that the arrival time of the last units to the destinations is four (i.e.  $c_{22}$  and  $c_{31}$  equal four) and all the other units of the commodity will arrive before time four. And therefore the closure is four and the minimal closure solution to this example problem is:

$$\begin{aligned}x_{11} &= 2 \\x_{22} &= 3 \\x_{23} &= 1 \\x_{31} &= 1 \\x_{32} &= 3\end{aligned}$$

and  $z = 4$ .

In the context of the Ships-to-Ports transportation problem, this minimal closure solution (i.e. solution to Bases-to-Ports transportation problem when formulated as a least-time transportation problem) is of great interest. This solution will render information concerning the arrival times of all the units of the supplies and it will also insure the arrival time of the last unit of the supplies at the ports will be minimal. The number of units transported and their corresponding arrival times at each port will be used in the Ships-to-Ports scheduling subproblem as input information. The quantity of supplies arriving at each port will dictate the number of ships that must travel to each port and the arrival times of the supplies will indicate the times at

when ships can be loaded and dispatched for the objective. In Chapter I it was stated that Ships-to-Ports scheduling problem could be formulated a maximal flow problem through a capacitated network. In Chapter III the generation of a network, which uses the information obtained from the optimal solution of the Bases-to-Ports subproblem, to accurately represent the Ships-to-Ports scheduling problem will be discussed.

### CHAPTER III

#### NETWORK CONSTRUCTION FOR THE SHIPS-TO-PORTS PROBLEM

The least-time transportation algorithm discussed in Chapter II, will generate the minimal closure solution to the Bases-to-Ports transportation subproblem. From the master problem's point of view, the supplies have been moved to the port in a minimal closure manner, now the ships must be scheduled to the ports so that they can be loaded with supplies and then travel to the objective area. If the Ships-to-Ports scheduling problem is to be formulated as a network flow problem, this network must incorporate information about the supplies and information about the ships in order to accurately represent the movement of both supplies and ships from the ports to the objective area. The supply information required is the quantities and arrival times of the supplies at each port. The network must reflect these arrivals through time. The ship information required is the arrival time of each ship to each port, if indeed every ship can travel to every port, and the travel time from each port to the objective area. The arrival time information for each ship would be known and available in a travel time matrix in which an entry would be the time required for that ship to travel to a particular port. The network must also reflect these arrivals through time. The port-to-objective area travel times are also known. Therefore, scheduling network must accurately represent all three pieces of information. To represent the arrival of supplies and ships and the departure of ships for the objective area, this

network must view each ports' operation through time. In addition the network must reflect each ships' travel to the scheduled port and from the port to the objective area. Conceptually, the construction of this network will take place in three steps, one for each of the necessary pieces of information.

Generally, the construction scheme might begin with the operation of each port through time. If each port was represented by a node of the network, the port's operation through time could be viewed by breaking up each port node into nodes to reflect the port at the end of a unit of time. Each of the subnodes would be connected by an arc which represents the passage of a unit of time at the port. The decomposition of the port nodes into subnodes and arcs connecting them is known as temporal expansion of a node. Each of these subnodes could represent a port at the beginning or the end of a time unit, but since there are not any supplies stored at the ports prior to transportation from the bases and there are not any ships docked at the ports at time zero, the subnode which might be the port at the beginning of the first time unit has no meaning. Additionally, the travel times for the supplies and the ships are at least equal to one time unit, the first unit of time passage at the port also is meaningless. Thus each subnode reflects a port at the end of a time period. There will be a temporal expansion for each of the ports in the Bases-to-Ports subproblem and its size (i.e. the number of time units represented) must be large enough to allow the arrival of all the supplies and the arrival of all possibly scheduled ships to that port.

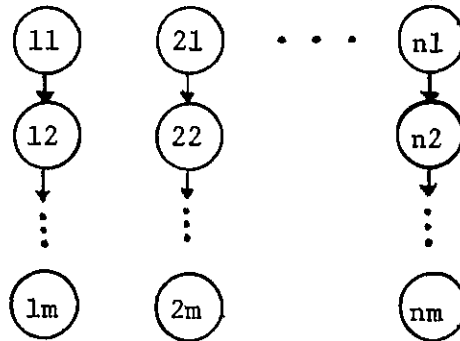
Each ship and its associated travel links to the ports must also be a part of this network. So each ship can be represented by a node, and from each ship node there must be arcs connecting it to the temporal expansions of the ports. These arcs will represent the travel links that a ship might travel over to reach a particular port. In general, there will be one arc emanating from a ship node for each port to which the ship might be scheduled. Each arc from a ship node must be connected to the particular subnode of the port's temporal expansion, which represents the time unit that the ship could arrive at the port if it were scheduled there.

A node must also be included in the network to represent the objective area. However, to simplify the actual construction of this network, let this node be decomposed into subnodes, known as port sinks. The objective area node will decompose into one port sink for each port in the Bases-to-Ports subproblem. Each port sink will be connected to the subnodes of its ports' temporal expansion by arcs which represent the travel link over which a ship must travel to reach the objective area. To complete this network and to facilitate the eventual solution of this network problem by the techniques of network theory, a super source node along the connecting arcs to the ship nodes, and a super sink node along with arcs from each of the port sink are added to the network.

This network, now known as the ship scheduling network, now will accurately represent the movements of ships to and from the ports and the arrival of supplies at the ports. A specific construction algorithm is now presented.

Ship Scheduling Network Construction Algorithm

STEP 1. Construct the temporal expansion of each port, where each node represents a port at the end of one time period, and each arc connecting the nodes represents the passage of a unit of time for that particular port.



where  $n$  = number of ports

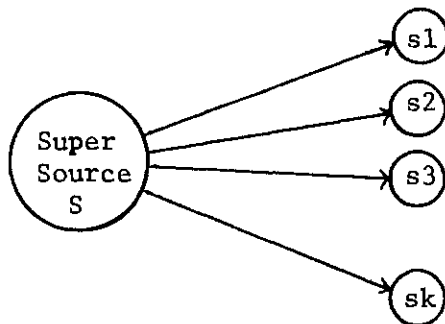
$$m = \max \{ t_{ij}, SP_{ij} \}$$

$$\forall j \in n$$

$t_{ij}$  = time at which supplies arrive at port  $j$

$SP$  = ship to port travel time matrix

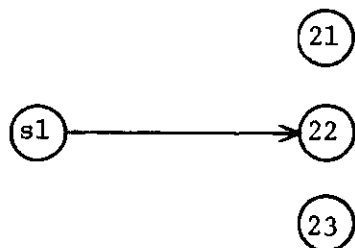
2. Create a super source node, and apply connecting arcs to a set of nodes which represents the ships that are to be scheduled.



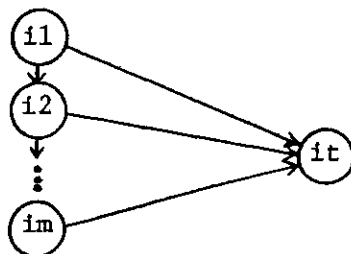
$K$  = Total number of ships available for scheduling.



3. Apply all connecting arcs from the ship nodes in Step 2 to the port nodes in Step 1 so that each pair of connected nodes will represent the time at which ship  $i$  can arrive at port  $j$ , e.g., if ship 1 cannot arrive at port 1 until time 2 then the node pair would be:

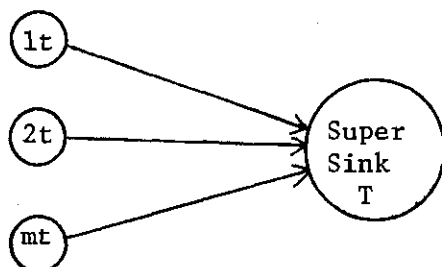


4. Create a sink node for each port  $i$ , and apply connecting arcs from each node in step 1 to the sink (i.e., there will be one arc to the sink node for each of the  $m$  nodes in the time expansion of port  $i$ ).

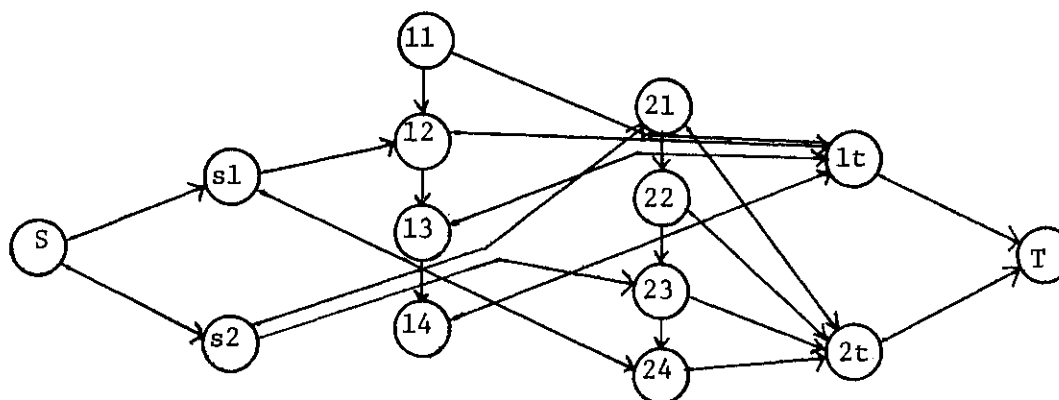


where the node is labeled  $(it)$  to represent the sink for port  $i$ .

5. Create a super sink node and apply connecting arcs from the port sink nodes,  $it$ , to the super sink node,  $T$ .



The complete network for a two ship, two port problem is shown below.



After constructing the ship scheduling network, the information gained from the optimal solution of the Bases-to-Ports transportation problem must be incorporated with it, so as to accurately represent the arrival of the supplies at the ports. The arrival of the ships must also be reflected on this network. In addition to the arrival of the supplies the actual quantities of supplies must be represented. Since the network will reflect the movement of both supplies and ships, a common unit of flow through the network was necessary. Hence, the assumption that supplies were transported in ship load quantities was made, but this assumption was reasonable in light of the 1980-1990 military resupply problem discussed in Chapter I. To accomplish the representation of arrivals of supplies and ships, lower and upper capacities will be applied to arcs of the ship scheduling network to reflect the requirements of flow through the network.

The amount of flow through the network from the super source to

the super sink will be a known, fixed amount equal to the number of ships available (which is equivalent to the total number ship loads of supplies that must be transported from the inland bases to the objective area). With the optimal solution to the Bases-to-Ports transportation subproblem, the assignment of the units of supplies to the ports is known. Each ship load of supplies requires a ship to transport it to the objective area. Thus, if  $\sum_i x_{ij}$  equals the number of ship loads of supplies that will eventually arrive at port  $j$ , it also is the number of ships that must move through port  $j$  in order that these supplies are transported to the objective area.

In general terms, the information from the optimal solution of the Bases-to-Ports subproblem will be used as upper and lower arc capacities of the ship scheduling network. The arcs representing the passage of a unit of time at a port will not have any unlimited storage of the supplies at the ports and there is an unlimited number of berths for the ships at the ports. Since the supplies at a particular port can be stored until a ship can arrive, the upper capacities are accumulative however, so that the upper capacity of port  $j$  at time  $p$ , is the sum of the ship loads that arrive at time 1 plus those that arrive at time 2, etc. (in general the upper capacity at port  $j$  in time  $p$ ,  $u_{jp} = \sum_{k=1}^m x_{ijk}$ , where  $x_{ijk}$  = ship loads of supplies that arrive from base  $i$  to port  $j$  at time  $k$ , and  $m$  is the total passage of time at the port in the temporal expansion). Due to the storage capability of the ports, it is unnecessary and sometimes not possible for a ship to enter the port for loading purposes exactly when the

supplies arrive, thus lower arc capacities on flow would have no meaning, so a zero value is used for these arcs.

The arcs which connect subnodes of ports' temporal expansion and the ports' port sink node represent the travel link between this port and the objective area. So at the end of  $p$  time units in port  $j$ 's operation, the amount of supplies that have arrived is  $\sum_{k=1}^p x_{ijk}$ , this number also represents the maximum number of ships that could have moved through port  $j$  and onto the objective area. Therefore, the upper capacities on these arcs must indicate this upper bound on the amount of flow at time  $p$ , so for an arc connecting a subnode at the end of time unit  $p$  to the port sink node will have an upper capacity of  $\sum_{k=1}^p x_{ijk}$ . There is not a required amount of flow on a particular arc of this type, so the lower capacity will be zero.

The total amount of supplies that move through port  $j$  is now  $\sum_k \sum_i x_{ijk}$ , and this number of ships must also move through port  $j$ . In order to require exactly  $\sum_k \sum_i x_{ijk}$  ships come to port  $j$  to take on supplies over the whole temporal expansion of port  $j$  a lower bound on the flow in some arcs must be used. Since the flow in the network (i.e. the ships) that has come through any of the nodes in the temporal expansion of a port must leave through the port sink node, then if a lower bound and an upper bound on arc capacity were placed on the arc between the port sink node and the super sink, an exact amount of flow (or ships) could be forced through the port. Therefore, a lower bound equal to  $\sum_k \sum_i x_{ijk}$  and an upper bound equal to  $\sum_k \sum_i x_{ijk}$  are placed on the arc to insure that  $\sum_k \sum_i x_{ijk}$  ships will move through port  $j$ .

(carrying  $\sum_k \sum_i x_{ijk}$  ship loads of supplies).

Bounds on arc flows must also be applied to those arcs which represent the travel links for the ships to the ports. Since each ship can only travel to at most one port, the upper bound on the flow in these arcs must be one. However, each ship can travel to anyone of the  $n$  ports, so the lower bound on these arcs must be zero. If the lower bound was one, the ship would be required to travel to all of the ports which would be equivalent to split the ship into  $n$  pieces, and each piece would have to travel a port. This situation would cause a discrepancy with the units of supplies at the ports, in that a ship load of supplies would be carried by only a portion of a ship. Explicit lower and upper bounds on flow are not necessary for the arc, connecting the super source node and the ship nodes, since they are only added to network to link up the super source node to the rest of the network. However, each unit of flow through network does represent a ship travel and since each ship can only be assigned to one port, an upper bound of one and a lower bound of zero will be used as the capacity restriction on these arcs.

With the use of lower and upper bounds on arc capacity, the actual arrival time of the ship loads of supplies is represented on the network. However, the travel times of ships to the ports and from the ports to the objective area must also be represented accurately since this network will become the input to the procedure which will find minimal closure assignment (or schedule), of the ships. The matrix of "ship to port" travel times ( $SP = [sp_{ij}]$ , where the  $ij^{th}$  element is the time required by ship  $i$  to travel to port  $j$ ) is known and elements of this

matrix will be applied as lengths to the arcs connecting the ship nodes to the port nodes (i.e.,  $(si) \xrightarrow{\text{LENGTH} = SP_{ij}} (jp)$ ). The vector of "port to objective" travel times ( $PO = [po_i]$ , where  $i^{\text{th}}$  element is the time required for any ship to travel from port  $i$  to the objective area) is also known and elements of this vector will be applied as lengths to the arcs connecting the port subnodes to their port sink node (i.e.,  $(ip) \xrightarrow{\text{LENGTH} = po_i} (it)$ ).

Algorithm for the Application of Lower and Upper Capacities, and  
Lengths to Arcs of the Ship Scheduling Network

The algorithm below summarizes the integration of the information obtained in the optimal solution of the Bases-to-Ports transportation problem with the travel time information to form an accurate representation, in network form, of the ship scheduling subproblem.

GIVEN: (1) optimal solution of the Bases-to-Ports transportation

problem (i.e.,  $x_{ij}$ 's and  $t_{ij}$ 's) where  $x_{ij}$  = quantity of supplies transported from base  $i$  to port  $j$

and  $t_{ij}$  = the arrival time, at port  $j$ , of supplies

$x_{ij}$

(2) matrix of "ship to port" travel times (i.e., SP matrix)

(3) vector of "port to objective" travel times (i.e., PO vector)

(4) constructed network from the above procedure

STEP 1: For all arcs in the temporal expansion of port  $j$  (i.e., arcs connecting port  $j$  in time period  $s$  to port  $j$  in time period

t) make the lower bound = zero, and the upper bound =  $\sum_{V_k} \sum_{V_i} x_{ijk}$  (or any large number since there is no restriction on supply storage at a port, and the length = one.

STEP 2: Apply lower bounds = zero and upper bounds =  $\sum_{k=1}^p x_{ijk}$ , to the arc between the port node in time period p and its port sink node. The length of each arc will be  $PO_j$  (i.e.,  $j^{\text{th}}$  element of PO vector).

STEP 3: For all the ports, make the lower bound = upper bound  $\sum_{V_k} \sum_{V_i} x_{ijk}$  for the arc connecting the sink node of port j to the super sink node and the length of each arc = zero.

STEP 4: For all the arcs connecting the ship nodes to the port nodes, make the lower bound = zero, the upper bound = one, and the length =  $SP_{ij}$  (e.g., the arc connecting ship i to port j has a length =  $SP_{ij}$ ).

STEP 5: All arcs from the super source to the ship nodes have a lower bound = one, an upper bound = one and a length = zero.

#### EXAMPLE

As an example, let us look at the constructed network on page 27. The arcs of the temporal expansion of the port would have lower bounds of zero and upper bounds equal to the sum of all the supplies arriving at that port. Thus if one ship load of supplies will arrive at port 1 then the upper bound on arcs (11,12) (12,13) and (13,14) would be one and the lower bound would all be zero. The length on each arc would be one to reflect the passage of one unit of time. If the one ship load of supplies arrives at time period two then the upper capacity on arc (11,1t) is zero, arc (12,1t) is one,

arc (13, 1t) is one, and arc (14, 1t) is one. All of the lower capacities are zero and the lengths equal the first element of the P0 vector, for these arcs. Arc (1t, T) will have upper and lower capacities of one since one ship is required to travel to port 1 and take on supplies. There will be a length of zero on arc (1t, T). Arc (S, S1) will have upper and lower capacities of one and a length of zero also. However, arc (S1, 12) will have a lower capacity of zero, an upper capacity of one and the length equal to the  $SP_{11}$  (i.e., travel time of ship one to port one).

#### Formulation of the Ships-to-Ports Subproblem

A network, including arc capacities and lengths, can be constructed to represent the Ships-to-Ports scheduling subproblem. This network combines the solution of the Bases-to-Ports subproblem and ship travel time information into a cohesive formulation, which then can have standard linear programming or network techniques applied to it. A mathematical formulation of the ship scheduling, in matrix form, is:

$$(1) \quad \text{Max } cx$$

s. t.

$$b^- \leq Ax \leq b^+$$

$$x \geq 0$$

where  $c$  is a vector of all 1's

$b^-$  is a vector of the lower arc capacities

$b^+$  is a vector of the upper arc capacities

$x$  is a vector variables, (i.e. flows on paths through



the network)

A is the arc-path incidence matrix, where the

$ij^{\text{th}}$  element is a one if path  $j$  uses arc  $i$ ; and

the  $ij^{\text{th}}$  element is zero otherwise.

This formulation indicates that the flow (number of ships) on paths through network is to be maximized, subject to the upper and lower capacities on arcs in the network. Problem (1) is the single commodity, maximal flow problem through a capacitated network. There are a number of algorithms available, such as OUT-OF-KILTER in Ford and Fulkerson (4), which can be used to solve this type of problem. However the objective of the Ships-to-Ports subproblem is not only to obtain a maximal flow (i.e. assure that all of the ships are scheduled through the ports), but also to obtain the minimal closure schedule of the ships (i.e. a schedule which minimizes the arrival time of the last ship at the objective area). Hinkle (1) has described an algorithm which will obtain both of these objectives in a network which has only upper capacity constraint, instead of both capacity constraints as in problem (1). His algorithm will be examined for its applicability to the Ships-to-Ports scheduling problem in the next chapter of this thesis.

## CHAPTER IV

### DISCUSSION OF HINKLE'S WORK

In his dissertation, Hinkle developed an algorithm, known as the min-max path flow algorithm, to solve a problem in which the maximal flow through an upper capacitated network is required and in addition, the length of the longest path carrying flow through the network must be minimal. As in Chapter II an example is now presented to indicate the difference between a minimal cost solution to the maximal flow network problem and the solution if the length of the longest path must be minimal in the maximal flow problem. Figure III shows a network used by Hinkle, in which all of the arc capacities are one and the cost (in this discussion cost and length are used interchangeably) of each arc is a positive number. If this network is examined, it can easily be seen that the maximal flow through it, from the source  $s$  to the sink  $t$ , is two. The minimal cost solution would use paths  $\{s, 1, 4, 5, t\}$  and  $\{s, 3, 6, t\}$ . The cost of path one is 6 and the cost of path two is 12, for a total minimal cost of 18. However path two has a length of 12, and therefore path two is the longest path carrying flow through network. If the paths  $\{s, 3, 4, 5, 6, t\}$  and  $\{s, 1, 2, 5, t\}$  were used instead to carry the two units of flow, the total cost would be 20, however, the length of the longest path (i.e.  $\{s, 3, 4, 5, 6, t\}$ ) carrying flow has been reduced to 11. As in the least time transportation problem there exists a trade-off between minimal cost and the

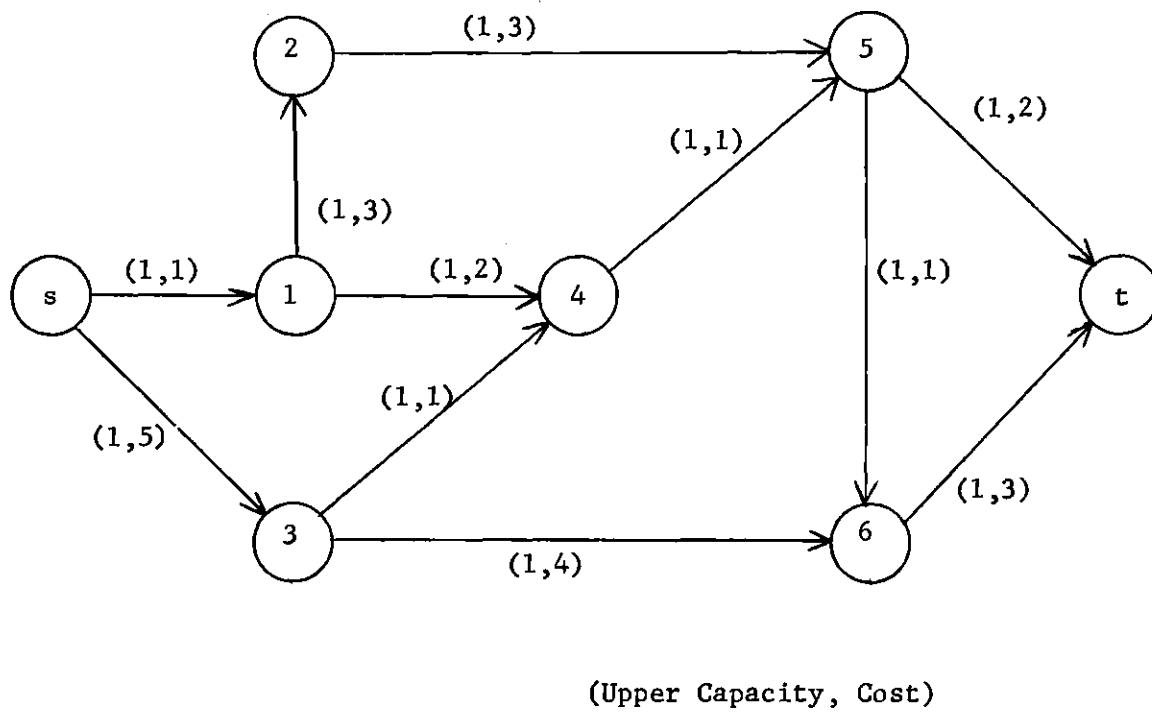


Figure 3. A Network with Only Upper Arc Capacities and Arc Costs.

length of the longest, flow carrying path through the network.

Since a standard transportation problem can be viewed as a simplified maximal flow problem through a capacitated network, the general algorithm discussed in Chapter II, for determining the least-time (min-max) solution to the Bases-to-Port transportation problem, could possibly be extended so that the min-max solution to a network problem could be found. The minimal cost flow network problem is analogous to the time-minimizing transportation problem in the context of a problem defined on a network like Figure III. The time-minimizing transportation problem found the set of travel links between the sources and destinations which minimized the total time required to transport all of the supplies. The minimal cost flow problem determines the set of paths, through the network over which the maximal amount of flow can be carried, such that the cost moving the flow through the network is minimized. Furthermore, if the arc costs are interpreted as the time required to traverse the arc, the minimal cost problem finds the set of paths which minimizes the total time required to move the flow through the network. A penalization scheme similar to the one used in Chapter II can also be used, although this is a bit trickier than in the case of the transportation problem. Instead of penalizing the links with the longest time carrying supplies, the path with the longest time carrying flow through the network will be penalized by assigning it an arbitrarily large time.

Hinkle indicates that one cannot simply assign penalties to the arcs in the longest path carrying flow, but must somehow penalize the path directly. One way to do this is to find a path with shorter

length, not presently in the solution, whose entry into the solution will cause flow to be removed from the longest path. We will explain how this can be achieved as the algorithm develops.

#### A General Min-Max Path Flow Algorithm

A general algorithm to determine the min-max path flow through a network will contain the same types of steps as the algorithm for obtaining the least-time solution to the transportation problem discussed in Chapter II along with the extension mentioned above.

STEP 1: Start with the initial arc cost values

STEP 2: Solve a minimal cost flow problem using the current path penalties

STEP 3: Determine if the optimal solution, in terms of the paths carrying flow, has changed from the previous solution. If YES, penalize the longest (highest cost) path carrying flow and return to STEP 2, If NO, Stop.

This algorithm will reduce the flow on the longest path in the network through the penalization scheme, since the minimal cost flow problem will attempt to find a set of paths which do not include any of the previously penalized paths. At each iteration the algorithm will find a new path or paths, on which the flow must be moved, which have length shorter than the length of the longest path in the previous solution to the minimal cost flow problem. Eventually this algorithm will not find a new path(s) of shorter length and will return the same solution as the one available at the previous iteration. Upon termination, the maximal flow will be carried on paths through the network such that the

length of the longest one is minimal (i.e. the optimal min-max path flow solution).

Hinkle (1) has developed an algorithm which uses the maximal flow solution to the network problem and a "reducing" path concept in order to get flow off the longest path. To reiterate, the min-max path flow problem determines the maximal flow through a capacitated network such that the length of the longest path carrying flow is minimal. Generally his algorithm begins with the maximal flow solution and then sequentially brings new paths into the solution set of the paths until either the flow on the longest path is reduced to zero and that path is dropped from solution set, or it is indicated that no further flow reduction is possible on the longest path. Since maximal flow through the network must be retained throughout the reduction process, there are three necessary conditions for a new path to enter the solution set. They are: (1) the path must maintain the maximal flow through the network; (2) the path must tend to reduce the flow on the longest path; and (3) the path must have shorter length than the longest path currently carrying flow through the network.

#### Determining Maximal Flow Through Capacitated Networks

The maximal flow problem through a capacitated network can be formulated in arc-path form as

$$\begin{aligned}
 (1) \quad & \text{Max} \quad cx \\
 & \text{s. t.} \quad Ax \leq b \\
 & \quad \quad x \leq 0
 \end{aligned}$$

The above formulation is a standard linear programming

problem in matrix form,

where  $c$  is a vector of all 1's

$b$  is a vector of arc capacities

$A$  is the arc path incidence matrix, a column of the  $A$ ,  $a_j$

is a vector which represents a path through the network

(i.e. the arcs used to make up the path)

and  $x$  is a vector of variables values (i.e. the flows on the paths of the network).

It must be noted that in the formulation the lower arc capacities are implicitly all zeros and the upper capacities are greater than or equal to zero.

A number of standard algorithms can be applied to problem (1) to obtain the maximal flow solution. Such algorithms as the primal simplex algorithm, or the revised simplex algorithm with an associated generation scheme for the entering variables. The primal simplex and revised simplex algorithm are described in both Tana (3) and Dantzig (2). A generation scheme for the revised simplex algorithm will determine a new path to enter the basis that will increase the value of the objective function. This new path will be a column vector consisting of zeros and ones, where a one in the  $i^{\text{th}}$  element of the vector indicates that the path uses the  $i^{\text{th}}$  arc of the network and a zero means it does not use that arc. One particular scheme is a shortest path algorithm which uses the current values of the dual variables as arc lengths, and thus the shortest path through the network becomes the entering path. This type of generation scheme is described in Ford and Fulkerson (4).

Finding a Path to Reduce the Flow on the Longest Path

Once having determined the maximal flow solution to problem (1) a path exhibiting the above three criteria must be found if the flow on the longest path current carrying flow is to be reduced. If the maximal flow through the network is to be maintained, the entering path must not reduce the total flow through the network. And clearly the path cannot increase the total flow, since this occurrence would violate that requirement of a maximal flow through the network. In linear programming terminology this entering path must generate an alternate optimal solution to problem (1) when it is brought into the basis. The L.P. entry criteria for a maximization problem is the relative cost coefficient of the new variable must have a negative value. Symbolically,  $z_j - c_j$ , where  $j$  is the index of the non-basic variable that will enter, must be negative. If the variable with the most negative value of  $z_j - c_j$  is brought into the basis, the largest increase in the objective function will occur. Thus if there were a large number of non-basic variables a minimization over the non-basic variables would be performed to determine the best variable to enter. This determination process can be expressed as;

$$(2) \quad \begin{array}{l} \text{Min } (z_j - c_j) \\ V_j, \text{ nonbasic} \end{array}$$

The minimal value of  $(z_j - c_j)$  will be zero since  $z_j - c_j$  must be greater than zero for all the non-basic paths or else optimality (maximal flow) would be contradicted.



The second criterion, which Hinkle has defined, is that the entering path must (tend to) reduce the flow on the longest path through the network. Let us determine what this criterion means in terms of linear programming. Let  $a_j$  be a vector of ones and zeros which is the original column of an entering variable and let  $B$  be the present basis matrix. Before the variable can be brought into the basis it must be updated by matrix multiplication with the basis inverse. This updating process results in the entering variable being represented as a linear combination of the columns of the basis. Let  $\hat{a}_j (= B^{-1}a_j)$  be the updated entering vector, and  $\hat{a}_j$  is brought into the basis through a standard pivot operation. If  $\hat{a}_j$  is to replace the  $i^{\text{th}}$  variable currently in the basis, then  $\hat{a}_{ij}$  must be positive. In the actual process of entering variable  $\hat{a}_j$ , all the elements  $\hat{a}_j$  must be brought to zero except  $\hat{a}_{ij}$ . This operation converts  $\hat{a}_j$  into a column of the basis. Therefore if an element of  $\hat{a}_j$  other than  $\hat{a}_{ij}$  is positive the value of the variable contained in that row of basis will decrease. To see this, let us examine the following diagram.

	$b$	$\hat{a}_j$
$k$	$b_k$	$\hat{a}_{ik}$
$i$	$b_i$	$\hat{a}_{ij}$

If  $\hat{a}_{ik}$  is positive, after entering  $\hat{a}_j$  the value of the  $k^{\text{th}}$  variable will be  $b_k - \frac{\hat{a}_{ik}}{\hat{a}_{ij}} b_i$ . Since  $b_i$ ,  $\hat{a}_{ij}$  and  $\hat{a}_{ik}$  are all positive the

the new value of the  $k^{\text{th}}$  variable,  $b'_k$ , will be less than  $b_k$ . So if there exists only one path at the longest length in the basis, and it is in the basis in row  $\ell$  then the entering updated path vector must have a positive value in the  $\ell^{\text{th}}$  element. If there happened to be more than one path in the basis of the longest length, the entering path must reduce the flow on this set of longest paths. Thus the sum of the updated vector elements corresponding to these paths in basis must be positive.

To affect the greatest reduction of flow on the set of longest paths in the basis, we would like to find the  $j^{\text{th}}$  nonbasis path which has the maximal sum of the elements in its updated vector corresponding to these longest paths. To find this new path, a search over the nonbasic path of the following form might be used;

$$(3) \quad \begin{array}{l} \text{Max} \\ v_j, \text{ nonbasic} \end{array} \quad \sum_{\ell \in \mathcal{L}} a_{\ell j}^*$$

where  $\mathcal{L}$  is the set of longest paths in the basis carrying flow. Since there is a minimization over the nonbasic paths in (2) and a maximization over the nonbasic paths in (3), these two searches could be combined into the following form;

$$(4) \quad \begin{array}{l} \text{Min} \\ v_j, \text{ nonbasic} \end{array} \quad p(z_j - c_j) - \sum_{\ell \in \mathcal{L}} a_{\ell j},$$

where  $p$  is a very large number. A large  $p$  is used so that the first term of this expression will dominate the second, that is the optimality of problem (1) must be assured. If a nonbasic path is found that has

the minimal value of this expression then that path will satisfy the first two of Hinkle's requirements for a path to reduce the flow on the longest path(s) in the basis carrying flow.

The last of Hinkle's conditions for a new path to enter the basis is that the length of this path be less than the length of the longest path(s). If a path with a shorter length is brought into the basis (i.e. the path also exhibits the other conditions) then some of the flow will be carried by this path and the min-max path flow solution will improve. Therefore the set of nonbasic valuables over which the minimization of (4) is conducted can be reduced to the set of nonbasic paths with a length less than or equal to the length of the longest path carrying flow minus one. Let  $L^*$  be the length of the longest path. By solving the following problem, the best nonbasic path  $j$  in terms of the three requirements, will be found.

$$(5) \quad \begin{aligned} & \text{Min} \quad p(z_j - c_j) - \sum_{v_l} a_{lj} \\ & \quad v_j, \text{ nonbasic} \\ & \text{s. t.} \quad L_j \leq L^* - 1 \end{aligned}$$

where  $L_j$  is the length of path  $j$ .

The objective function of problem (5) can be simplified, since

$$z_j = C_B B^{-1} a_j$$

where  $C_B$  is a vector of the cost coefficients of the basic valuables of problem (1).

And further,

$$z_j = C_B B^{-1} a_j = \pi a_j$$

where  $\pi (= C_B B^{-1})$  is the vector of dual variables for problem (1). Since the cost coefficient for a path variable of problem (1) is one, the first term of the objective function in (5) becomes

$$p(z_j - c_j) = p(\pi a_j - 1).$$

If we now turn our attention to the second term,  $\sum_{\forall \ell} a_{\ell j}$  can also be simplified. Since the updated entering vector is determined by matrix multiplication of entering vector and the basis inverse,  $a_j = B^{-1} a_j$ . The sum of certain elements of  $a_j$  is required, and thus  $\sum_{\forall \ell} a_{\ell j} = \sum_{\forall \ell} B_{\ell}^{-1} a_j$ . The objective function of (5) can now be written as,

$$\text{Min}_{j, \text{ nonbasic}} \quad p(\pi a_j - 1) - \sum_{\forall \ell} B_{\ell}^{-1} a_j$$

However, the minimization of the first term is equivalent to the minimization of  $p \pi a_j$  since the one is a constant. By combining terms in the objective function problem (5) can be written as

$$\begin{aligned} (6) \quad & \text{Min}_{j, \text{ nonbasic}} \quad (p \pi - \sum_{\forall \ell} B_{\ell}^{-1}) a_j \\ & \text{s. t.} \quad L_j \leq L^* - 1 \end{aligned}$$

To summarize briefly, Hinkle has defined three criteria for a path enter the basis of the maximal flow solution to a network problem if the flow on the longest path is to be reduced. Problem (6) is the mathematical formulation of a problem whose solution will be a path through the network that will satisfy the necessary criteria. (6) can

also be viewed as determining the path through the network which minimizes the sum of the arc costs of the form  $p_{\pi} - \sum_{\ell} B_{\ell}^{-1} v_{\ell}$  along it and its length is less than a known constant,  $L^* - 1$ . From this point of view the problem becomes a constrained shortest path problem. The general constrained shortest path problem has been formulated by Berry and Jensen (6) and Hinkle (1). A solution algorithm has been discussed by Hinkle and it applies to the arc costs,  $p_{\pi} - \sum_{\ell} B_{\ell}^{-1} v_{\ell}$ , and the known arc lengths to the network, and uses a labelling scheme to determine the optimal solution (i.e. the entering path) of problem (6). His algorithm requires that all of the arc costs are positive to avoid the difficulties arising from a network that could contain negative cost-directed cycles in it. If a cycle existed in the network with negative arc costs, his labelling routine would proceed around the cycle endlessly and if a path including this cycle was ever returned by problem (6), the value of the objective function would be negative infinity. Therefore if the arc costs are not positive, problem (6) can have an unbounded solution since Hinkle does not require the network to be acyclic (i.e. a directed network which contains no cycles). It is easy to see that negative arc costs would not be possible. From the dual of the maximal flow problem,

$$\begin{aligned}
 &\text{Min } \pi b \\
 &\text{s. t. } \pi A \geq C \\
 &\pi \geq 0
 \end{aligned}$$

We know all of the dual variables ( $\pi_i$ 's) must be positive in the optimal solution. Since Hinkle's min-max path flow algorithm begins

with the optimal solution to the maximal flow problem and  $p$  can be any positive number, by simply increasing  $p$  until  $p\pi > \sum_{\forall \ell} B_{\ell}^{-1}$ , all of the arc costs are assured to be positive.

Let us now summarize Hinkle's algorithm for determining the min-max path flow solution for a network which has only upper capacities on the arcs.

STEP 1: Obtain the maximal flow solution to problem (1) for the given network. Determine the set,  $\ell$ , of the longest paths carrying flow through the network and the value of  $L^*$ .

STEP 2: Using the current values of the dual variables,  $\pi$ , and the current basis inverse matrix,  $B^{-1}$ , determine the arc cost of the form  $p\pi - \sum_{\forall \ell} B_{\ell}^{-1}$ .

STEP 3: Obtain the optimal solution to the constrained shortest path problem (6) using the arc costs and lengths. If no path is found with a length less than the current  $L^*$  stop, otherwise continue to step 4.

STEP 4: Enter the new path by the standard linear programming techniques and determine the new length value,  $L^*$ , of the longest path carrying flow; return to step 2.

Hinkle has shown that his algorithm will converge to an optimal solution in a finite number of steps. Upon termination this algorithm will have determined the optimal min-max path flow solution for the given network, that is, the maximal amount of flow will be carried through the network on a set of paths such that the length of this longest path in this set is minimal.

Hinkle's algorithm for determining the min-max path flow solution would generate an optimal solution to the Ships-to-Ports subproblem formulated as problem (1) in Chapter III except that this problem has both lower and upper arc capacities and his algorithm in its present form will only solve the problem for a network with upper arc capacities. However, if his algorithm could be extended so that it would be applicable to the ship scheduling network developed in Chapter III it would determine the schedule of the ships to the ports such that the arrival time of the last ship at the objective area would be minimal. In addition this schedule would insure that the last unit of the supplies would also arrive in the minimal amount of time.

Extensions of Hinkle's Algorithm for  
the Ships-to-Ports Subproblem of Chapter III

Let us examine problem (1) in more detail to ascertain what extensions of Hinkle's algorithm are required before it is capable of solving the Ships-to-Ports scheduling problem. Recall the mathematical formulation of the ship scheduling problem is,

$$\begin{aligned}
 (7) \quad & \text{Max } cx \\
 & \text{s. t.} \\
 & b^- \leq Ax \leq b^+ \\
 & x \geq 0
 \end{aligned}$$

and the dual of problem (7) is,

$$\begin{array}{ll}
\text{Min} & \pi^- b^- + \pi^+ b^+ \\
\text{s. t.} & \pi^+ A \geq C \\
& \pi^- A \geq C \\
& \pi^+ \geq 0 \\
& \pi^- \leq 0
\end{array}$$

where  $\pi^+$  is the vector of dual variables corresponding to the upper capacity constraints and

$\pi^-$  is the vector of dual variables corresponding to the lower capacity constraints.

Hinkle's algorithm begins with the maximal flow solution to the network problem. The maximal flow problem in a network which has both lower and upper arc capacities, as in problem (7) would require a basis of size  $2m$ , where  $m$  is the number of arcs in the network. The basis size could be reduced if some of the lower arc capacities are zero, since the constraints corresponding to these zero lower capacity arcs would be linearly dependent and redundant with respect to the non-negativity constraints for all the path flow variables in the primal problem (7). In general though, this type of basis size reduction would not be possible and there would be two rows and two variables in the basis for each constraint of the maximal flow problem. Thus the basis size would severely restrict the size of the networks for which Hinkle's algorithm would be applicable. However, both constraints for a particular arc cannot be binding at the same time. To see this fact, let us look at an example of a possible constraint in problem (7),

$$b_i^- \leq \sum_{j=1}^m a_{ij} x_j \leq b_i^+$$



and specifically the constraint

$$2 \leq x_1 + x_2 + x_3 \leq 10$$

if  $i^{\text{th}}$  arc had an upper capacity of 10 and a lower capacity of 2, and there are three paths using it. Clearly it would be impossible for the sum of the flows on the three paths to simultaneously equal 10 and 2. Furthermore only one of the arc capacities could bind the flow on all of the paths using the arc. An algorithm is discussed in Chapter V which will take advantage of this fact in both the maximal flow problem and the entering of a reducing path steps of Hinkle's algorithm.

Let us now examine the constrained shortest path generation scheme in the context of the maximal flow solution to a network with both upper and lower arc capacities. In the optimal solution to the maximal flow problem it is possible for some of the dual variables to be negative. It can be seen from the dual of problem (7), that if any of the  $\pi^-$  variables are in the optimal basis of the dual, they must have a negative value. Let us now partition the vector of dual variables for the optimal solution to the maximal flow problem into two sets, the upper constraint dual variables  $\pi^+$  and the lower constraint dual variables  $\pi^-$ . Hence  $\pi$  can be expressed in vector notation as  $[\pi^+, \pi^-]$ . The objective function of the constrained shortest path problem (6) is

$$(p - \sum_{\ell \in V} B_{\ell}^{-1}) a_j$$

In particular, let us examine the vector of arc costs,

$$P\pi = \sum_{\forall \ell} B_{\ell}^{-1}$$

If the partitioning of the  $\pi$  vector is applied to this expression for the arc costs, they can be divided into 2 sets. The first set

$$(8a) \quad P\pi_i = \sum_{\forall \ell} B_{\ell i}^{-1}, \quad \pi_i \in \pi^+$$

contains the costs for arcs which have the upper capacity constraint in use in the basis, and this constraint does not necessarily have to be binding. Under an extended entry rule, Hinkle has shown for any arc  $i$  and any row  $B^{-1}$  of  $B^{-1}$  that if  $\sum_{\forall \ell} B_{\ell i}^{-1} > 0$  that  $\pi_i > 0$ . Since  $p$  can be any arbitrarily large positive number, the arc costs of set (8a) can always be made positive if  $\sum_{\forall \ell} B_{\ell i}^{-1}$  is greater than zero. The second set

$$(8b) \quad P\pi_i = \sum_{\forall \ell} B_{\ell i}^{-1}, \quad \pi_i \in \pi^-$$

contains the costs for arcs which have the lower capacity constraint in use in the basis, and again this constraint can be either binding or not. If a constraint corresponding to an arc of this set is binding then the dual variable must be either zero or negative in value. If the dual variable is zero there does not exist any difficulties with the arc cost. However, if the dual variable is negative the  $p\pi_i$  term of the arc cost expression will dominate the  $\sum_{\forall \ell} B_{\ell i}^{-1}$  term, and it is possible to obtain a negative arc cost.

For a general network, either cyclic or acyclic, the constrained shortest path problem requires that all arc costs are positive in order to avoid an infinite solution that could be generated if a negative cost,

directed cycle existed. If the constrained shortest path problem is to be used to generate a path to reduce the flow on the longest path in a network which has both upper and lower arc capacities, negative cost cycles must still be avoided. Since there exists a real possibility that some arc costs are negative, can negative cost directed cycles be avoided. Recall the network, whose construction was described in Chapter III, that will be used in the Ships-to-Ports subproblem. In this network ships (units of flow) move from the source to the temporal expansion of a port and then on to the port's sink, which represents the objective area. Since the temporal expansion of a port was constructed in network form to represent a port's operation through time, it would be illogical to have cycles. A cycle in the temporal expansion would indicate that a ship could arrive at a port, go through the cycle and depart from the port at a time before it arrived. Therefore each port's temporal expansion must acyclic. It is also impossible for a ship to be scheduled to more than one port and thus there are no arcs between ports. So it is not possible for a ship to cycle between temporal expansions of different ports. Hence the complete portion of the ship scheduling network representing ports is acyclic. All other arcs of the network are forward directed arcs, either from ship nodes to the ports portion of the network or from the ports to the port-sink nodes. It can be seen, therefore, that the ship scheduling network, for the Ships-to-Ports subproblem, is completely acyclic. So even though negative arc costs are possible, no negative cost-directed cycles can ever exist. In Chapter VI, a modification of Hinkle's labeling algorithm is presented, which takes advantage of this acyclic property

of the network for the Ships-to-Ports scheduling problem, for the solution to the constrained shortest path problem.

## CHAPTER V

## CONSTRAINT SWITCHING

The maximal flow problem with lower and upper bounds can be formulated as:

$$\begin{aligned} \text{Max } z &= cx & (1) \\ \text{s. t. } b^- &\leq Ax \leq b^+ \\ x &\geq 0 \end{aligned}$$

where  $c$  is a vector of all 1's.

$b^-$  and  $b^+$  are the vectors of lower and upper bounds on the flows in the arcs,

$x$  is a vector of the flows on the paths, and

$A$  is the arc-path incidence matrix ( $a_{ij} = 1$ , if path  $j$  uses arc  $i$ ; and  $a_{ij} = 0$ , otherwise).

Decomposing the constraint in (1), and rewriting (1) in equation form, we obtain

$$\begin{aligned} \text{Max } z &= \sum_j x_j & (2) \\ \text{s. t. } \sum_j a_{ij} x_j + s_i &= b_i^+ & \forall i & (A) \\ \sum_j a_{ij} x_j - t_i &= b_i^- & \forall i & (B) \\ x_j, s_i, t_i &\geq 0 & \forall i \text{ and } \forall j \end{aligned}$$

The SUBOPT Procedure Applied to the Maximal Flow Problem

A number of methods have been proposed to solve this problem in either of the above forms. Robers and Ben-Israel (5) have developed a procedure called SUBOPT (sub-optimization method) to solve problem (1), which they refer to as an interval programming problem. SUBOPT decomposes the master problem (1) into a subproblem of the special form

$$\text{Max } z = cx \quad (3a)$$

$$\text{s.t. } b^- \leq Bx \leq b^+ \quad (3b)$$

$$b_s^- \leq a_s x \leq b_s^+ \quad (3c)$$

where B is a nonsingular, nxn matrix,

n is the number of variables in the master problem,

$a_s$  is the  $s^{\text{th}}$  row of the original A matrix,

$\begin{bmatrix} B \\ a_s \end{bmatrix}$  is a submatrix of A

and  $b^- \leq b^+$  and  $b_s^- \leq b_s^+$  are the lower and upper bounds on the constraints from the master problem.

In order to solve the above subproblem a variable change is made.

$$Y = Bx$$

$$x = B^{-1}Y$$

Subproblem (3) is now written as

$$\text{Max } z = cB^{-1}y \quad (3a')$$

$$\text{s.t. } b^- \leq Y \leq b^+ \quad (3b')$$

$$b_s^- \leq a_s B^{-1} \leq b_s^+ \quad (3c')$$

If the optimal solution to (3') is  $Y^*$ , then the optimal solution to (3) is  $x^* = B^{-1}Y^*$ .

At each iteration, SUBOPT solves a subproblem of the form (3) where the B matrix is formed by using n linearly independent rows of the original A matrix and the constraint (3c) is one of the other (m-n) constraints. Instead of solving this subproblem directly, a variable change is performed to obtain the subproblem (3'). This new subproblem is solved by initially assigning each variable the corresponding lower or upper b value depending upon the cost coefficient of the variable in the objective function. If the initial solution does not satisfy the constraint (3c') then this solution altered in such a way that will guarantee that all of the constraints (both 3b' and 3c') are satisfied and the decrease in the value of the objective function due to this alteration of the initial solution is minimal. Once having obtained an optimal solution to subproblem (3'), then an optimal solution to subproblem (3) is also obtained. This solution is checked for feasibility in the other (m-n-1) constraints of the master problem. If any of the constraints are violated a new subproblem of the form (3) is solved using one of the violated constraints as the (n+1)<sup>st</sup> constraint.

We continue to solve these subproblems at each iteration until at some point either one of the subproblems is infeasible, implying master problem (1) is infeasible, or the solution to the subproblem

satisfies all of the constraints of the master problem, and then we can terminate the SUBOPT procedure. The finiteness of this procedure has been shown by Robers and Ben-Israel in (5). This SUBOPT will solve problem (1) and render either an optimal solution or the information about the infeasibility of the problem in a finite number of iterations.

However, there are a number of disadvantages in applying the SUBOPT procedure to a maximal flow network problem with lower and upper bounds on flows in arcs. First, all  $n$  variables are required to be known explicitly in the problem for both the variable change (i.e.,  $(Y=Bx)$ ) and the size of the  $B$  matrix. Since the maximal flow problem is in arc-path formulation the knowledge of all the variables would require that all the possible paths through the network be enumerated. For even moderately large networks this task would be extremely inefficient, if it could be accomplished at all. Secondly, SUBOPT requires the number of variables  $n$  (i.e., the number of paths) to be less than the number of constraints  $m$  (i.e., the number of arcs). Since there is one variable for each path and one constraint for each arc in the network, SUBOPT requires the number of paths to be less than the number of arcs. However, in a general network the maximum possible number of directed arcs is  $x(x-1)$  where  $x$  is the number of nodes in the network, and the maximum number of paths is  $2^{y-x+1}$ , where  $y$  is the number of directed arcs. Hence, for a general network this requirement would not be fulfilled. However, SUBOPT could still be applied to networks where the number of paths is less than the number of arcs, but this requirement severely limits the number of networks that could be solved by SUBOPT. Thirdly, a series of subproblems must be solved in order to obtain an



optimal solution to the master problem (i.e., problem (1)). Each of these subproblems requires a basis and basis inverse of size  $n \times n$ , where  $n$  is the number of variables, plus the  $(n+1)^{\text{st}}$  constraint which is used for a feasibility check (i.e., the constraint set for each subproblem contains  $n+1$  constraints). Thus each subproblem would require large amounts of storage for computer implementation due to the size of the basis and basis inverse. Fourthly there would be a large number of iterations of the SUBOPT procedure since each subproblem has only one constraint in the constraint set replaced by one of the  $(m - (n+1))$  other constraints of the master problems, which are not used in the solution of the subproblem. SUBOPT does, however, allow a smaller maximum number of possible iterations (i.e.,  $\binom{m}{n+1} = \frac{m!}{(n+1)!(m-n-1)!}$ ) than a standard linear programming problem. This reduction does not seem to outweigh the necessity of solving a complete subproblem at each iteration.

#### The Standard Linear Programming Approach to the Maximal Flow Problem

It would seem that standard linear programming could be applied to the maximal flow network problem with lower and upper bounds on the flow in the arcs, as formulated in (1). If problem (1) was rewritten as (2) and formulated as a linear programming problem, a basis matrix of size  $(2m \times 2m)$  would be required since each of  $m$  constraints in problem (1) decomposes into one constraint of type A and one constraint of type B. Thus linear programming would require two constraints for each arc of the network and thus large networks would require that  $2m$  variables be in the basis. Hence, the  $m$  slack variable from the

constraints of type A would be used. However, the slack variables from the constraints of type B could not be used in the basis since they would have a value less than zero (i.e.,  $-t_i = b_i^- \Rightarrow t_i = -b_i^-$ ), which violates the nonnegativity constraint. Therefore,  $m$  artificial variable must be added to the slack variables from the type A constraints (i.e.,  $s_i$ 's) to complete  $2m$  variables needed for the initial basic solution. Once these artificial variables have been introduced into the problem, formulation (2) becomes:

$$\text{Max } z = \sum_{\forall j} x_j - \sum_{\forall i} Mp_i \quad (4)$$

s. t.

$$\sum a_{ij}x_j + x_i = b_i^+ \quad \forall i \quad (A)$$

$$\sum a_{ij}x_j - t_i + p_i = b_i^- \quad \forall i \quad (B)$$

$$x_j, s_i, t_i, p_i \geq 0 \quad \forall i \text{ and } \forall j$$

where  $p_i$  is the artificial variable from the  $i^{\text{th}}$  constraint of type B and  $M$  is a large positive cost, if the big "M" method is to be employed to remove the artificial variables from the basis. These artificial variables will be removed first from the basis since linear programming uses  $\min \left\{ \frac{b_i}{a_{ik}} ; a_{ik} > 0 \right\}$ , where  $a_k$  is the entering variable, as an exit criteria and  $b_i^- \leq b_i^+$ ,  $\forall i$ . Hence, there must be  $m$  iterations before all the artificial variables are removed from the basis. Thus there seems to be some unnecessary inefficiency incurred by the use of linear programming on problem (2) as rewritten in (4). Another major limitation would be storage on the computer due to the large basis size

as indicated above, if implementation on the computer was necessary.

Theory of Constraint Switching in the Problem of  
Maximal Flow Through a Network with  
Lower and Upper Arc Capacities

Due to the disadvantages incurred by using SUBOPT or standard linear programming to problem (2), it was conjectured that this problem could be solved by treating only one type of constraint, either type A or type B, at a time and solving the resultant subproblem by standard linear programming. Thus, a problem as shown below could be solved.

$$\begin{aligned} \text{Max } z &= \sum x_j & (5) \\ \text{s. t. } \sum a_{ij}x_j + s_i &= b_i^+ \quad \forall i \\ x_j, s_i &\geq 0 \quad \forall i \text{ and } \forall j \end{aligned}$$

Once having obtained an optimal solution to problem (5), the dual variables could be used to determine which constraints were actually binding (i.e., if the  $i^{\text{th}}$  dual variable,  $\pi_i$ , had a value greater than zero then the slack variable corresponding to the  $i^{\text{th}}$  constraint,  $s_i$ , would have a value of zero and the  $i^{\text{th}}$  constraint would be an equality of the form  $\sum a_{ij}x_j = b_i^+$ . Thus the constraint will bind the variables  $x_j$ , to exact values). A subset of the  $m$  constraints that are binding could be determined. Let

$$B_5 = \{i \mid \pi_i > 0\} = \text{set of indices of the binding constraints}$$

$$B'_5 = \{i \mid \pi_i = 0\} = \text{set of indices of the non-binding constraints}$$

Clearly  $B_5 \cup B'_5 = m = \text{set of all } m \text{ indices}$ . The other subproblem of (2) could also be solved, namely

$$\begin{aligned}
 & \text{Max} \quad z = \sum x_j \\
 & \text{s. t.} \\
 & \quad \sum a_{ij} x_j - t_i = b_i^- \quad \forall i \\
 & \quad x_j, t_j \geq 0 \quad \forall i \text{ and } \forall j
 \end{aligned} \tag{6}$$

and the sets of binding and non-binding constraints would be

$$\begin{aligned}
 B_6 &= \{i \mid \pi_i > 0\} \\
 B'_6 &= \{i \mid \pi_i = 0\}.
 \end{aligned}$$

Since only the constraints of  $B_5$  are required to bound the solution space of problem (5) and the constraints of  $B_6$  are necessary to the solution space of problem (6) it would seem that the union of sets  $B_5$  and  $B_6$  would be the only constraints necessary to completely bound the solution space of problem (2). However, in general  $B_5 \cup B_6 \neq \{m\}$  i.e., the union of the two sets of binding constraints would not equal the set of binding constraints for problem (2). However, this decomposition of problem (2) brought about the discussion of another way to look at the constraint set of (2) and sufficient condition for a binding set of constraints in Lemma 1.

In formulation (2) consider the  $i^{\text{th}}$  constraint in set (A) and its associated constraint in (B). This pair can be expressed as either

$$\sum a_{ij}x_j + s_i = b_i^+$$

$$0 \leq s_i \leq b_i^+ - b_i^-$$

or

$$\sum a_{ij}x_j - t_i = b_i^-$$

$$0 \leq t_i \leq b_i^+ - b_i^-.$$

$s_i = b_i^+ - b_i^-$  (its upper bound) implies the lower constraint  $\sum a_{ij}x_j - t_i = b_i^-$  is binding, and similarly for  $t_i = b_i^+ - b_i^-$ . Thus, by introducing upper bounds on variables (slacks) we can eliminate one of the two constraints. Further, to eliminate the need for keeping track of variables at their upper bounds we will develop a technique of constraint switching when a variable reaches its upper bound.

We state as a lemma below, that there can be only  $m$  constraints in the constraint set at any particular iteration and furthermore that for each arc  $i$  there will be only one constraint in the set of either type A or B.

Lemma 1 At any particular iteration,  $k$ , one and only one constraint will be active for each arc in the network, and this constraint will be of type A or B.

Proof: Let  $x_{B_i}$  be the flow on arc  $i$  at some iteration  $k$ , then  $x_{B_i}$  is feasible in terms of problem (1), if  $b_i^- \leq x_{B_i} \leq b_i^+$ .

Case 1 If  $x_{B_i}$  is increasing we wish to bound the flow on arc  $i$  at  $b_i^+$ . If  $x_{B_i} = b_i^+$  we know that

$$\sum_j a_{ij}x_j = b_i^+$$

which implies  $s_i = 0$ .

We know that constraint  $A_i$  is binding in the sense that  $s_i = 0$  and thus  $A_i$  must be active in the constraint set in order that  $x_{B_i}$  remains feasible. We also know that  $t_i \neq 0$  and that  $\sum_j a_{ij}x_j > b_i^-$  so that constraint  $B_i$  is not binding and is unnecessary to the feasibility of  $x_{B_i}$ .

Case 2 If  $x_{B_i}$  is decreasing we wish to bound the flow on arc  $i$  at  $b_i^-$ . If  $x_{B_i} = b_i^-$  we know that

$$\sum_j a_{ij}x_j = b_i^-$$

which implies  $t_i = 0$ .

Again we see that constraint  $B_i$  is binding and that  $B_i$  must be active in the constraint set so that  $x_{B_i}$  remains feasible. Again we also know that  $s_i \neq 0$  and  $\sum_j a_{ij}x_j < b_i^+$ , so that  $A_i$  is not binding and is unnecessary to the feasibility of  $x_{B_i}$ .

Case 3 Since the flow on arc  $i$  is bounded by  $b_i^-$  and  $b_i^+$ , the slack variables  $t_i$  and  $s_i$  can be viewed as indicating the distance which the flow is from the lower and upper bound. Thus if the flow on arc is not at either bound, the slack variables have value (i.e.,  $t_i, s_i \leq b_i^+ - b_i^-$ ).

Therefore either constraint  $A_i$  or  $B_i$  can be used to express the flow requirements on arc  $i$ , since they are equivalent, as seen below.

$$A_i: \sum_j a_{ij}x_j + s_i = b_i^+$$

$$B_i: \sum_j a_{ij}x_j - t_i = b_i^-$$

$$(A_i - B_i) s_i + t_i = b_i^+ - b_i^-$$

and since  $b_i^+ - b_i^-$  is the total amount of slack variable at arc  $i$ , it can be seen that constraint  $A_i$  can be obtained from  $B_i$  by substituting  $t_i = b_i^+ - b_i^- - s_i$  into  $B_i$ .

$$\sum_j a_{ij}x_j - (b_i^+ - b_i^- - s_i) = b_i^-$$

$$\sum_j a_{ij}x_j + s_i = b_i^+ \text{ which is } A_i$$

Therefore, we only need one constraint, either  $A_i$  or  $B_i$  to express the flow requirements on arc  $i$ .

In fact, for this case we actually don't need either constraint.

There are three different variables that can be in the basis.

They are a lower-bound slack  $t$ , an upper-bound slack  $s$ , and a path  $x$ .

The bounds on these variables are

$$0 \leq t_i \leq b_i^+ - b_i^- \quad \forall i$$

$$0 \leq s_i \leq b_i^+ - b_i^- \quad \forall i$$

$$0 \leq x_j \leq \infty \quad \forall j$$

A variable can enter the basis of a linear programming problem, under maximization of the objective function, if

$$z_j - c_j < 0$$

$$c_B B^{-1} a_j - c_j < 0$$

where  $a_j$  is the entering vector

$c_j$  is the coefficient in the objective function

$$c_j = \begin{cases} 1 & \text{if entering vector is a path} \\ 0 & \text{if entering vector is a slack} \end{cases}$$

$$\left| z_k - c_k \right| = \max_{\forall j \notin \text{basis}} \left| z_j - c_j \right|$$

The above expression is used as the entry criterion to determine the variable  $a_k$  to enter the basis. As the flow on (value of)  $a_k$  increases two things can happen:

- (1) one of the variables reaches its lower bound first (i.e., zero) and an ordinary simplex pivot operation is performed.
- (2) one of the slack variables reaches its upper bound first, (i.e.  $b^+ - b^-$ ) and a constraint switch is performed, followed by a simplex pivot operation.

To ascertain which of the two cases above happens, we must determine the blocking variables. To find blocking variable let

$$\theta_1 = \min_i \left\{ \frac{\hat{b}_i}{\hat{a}_{ik}} \mid \hat{a}_{ij} > 0 \right\}$$

$$\theta_2 = \min_i \left\{ \frac{(\hat{b}_i^+ - \hat{b}_i^-) - \hat{b}_i}{-\hat{a}_{ik}} \mid \hat{a}_{ik} < 0 \right\}$$



only if  $i^{\text{th}}$  variable is a slack

$\theta_3 = u_k =$  amount of flow possible on entering  
vector

where  $\hat{b}_i = i^{\text{th}}$  element of the updated b-vector

$$\hat{b} = B^{-1}b$$

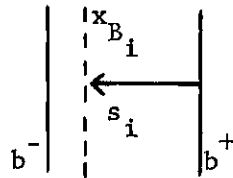
$\hat{a}_{ik} = i^{\text{th}}$  element of the updated entering vector

$$a_k(\hat{a}_k = B^{-1}a_k)$$

The blocking variable is associated with the constraint for which

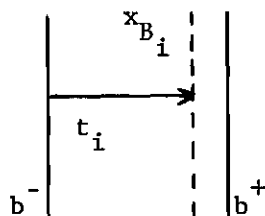
$$\theta = \min \{\theta_1, \theta_2, \theta_3\}$$

Once the blocking variable has been determined we have either case (1) or (2) above and we wish to exit this blocking variable from the basis. We want to exit the variable which will become infeasible first. If  $\theta = \theta_1$ , we have the standard linear programming exit procedure. However, if  $\theta = \theta_2$ , we have the standard linear programming exit procedure. However, if  $\theta = \theta_2$  the  $i^{\text{th}}$  slack variable is increasing and as shown in the diagrams below this will cause the flow on the  $i^{\text{th}}$  arc to exceed its bounds. If  $s_i$  is in the basis, it will increase causing the



flow  $x_{B_i}$  on the arc to decrease below  $b^-$ . If  $t_i$  is in the basis, it

will increase causing the flow,  $x_{B_i}$



to increase above the upper bound  $b^+$ . Therefore a constraint switch must be in order to bound the flow within  $b^+ - b^-$ .

#### Constraint Switching Algorithm

To perform a constraint switch:

(1) Exchange variables in the basis by multiplying the  $i^{\text{th}}$  column by  $-1$ .

$$a_i^T = [0, 0, 0, \dots, \ell_i, \dots, 0, 0] \text{ where } \ell_i = \begin{cases} 1 & \text{if } s_i \text{ in basis} \\ -1 & \text{if } t_i \text{ in basis} \end{cases}$$

Thus, if  $s_i \rightarrow t_i$ :  $B_{ii}: +1 \rightarrow -1$

$t_i \rightarrow s_i$ :  $B_{ii}: -1 \rightarrow +1$

$B_{ii}$  is the  $i^{\text{th}}$  element in column  $i$  of the current basis.

(2) If we now take the inverse of the basis the  $i^{\text{th}}$  row will be the negative (i.e.,  $-1$  times) of the  $i^{\text{th}}$  row before the variable exchange of (1). So we could just multiply the  $i^{\text{th}}$  row of the basis inverse to accomplish the exchange in the inverse.

(3) Since we have multiplied the  $i^{\text{th}}$  row of the basis inverse by  $-1$ , the  $i^{\text{th}}$  element of both  $\hat{b}$  (updated  $b$ -vector) and  $\hat{a}_k$  (updated

entering vector) will also be multiplied by -1. Thus the new  $i^{\text{th}}$  element  $\hat{a}_k$  is  $-\hat{a}_{ik}$  and the new  $i^{\text{th}}$  element of the b-vector is  $-\bar{b}_i + (b_i^+ - b_i^-)$ .

(4) Now an ordinary simplex pivot is performed with  $\hat{a}_k$  entering the basis to replace the  $i^{\text{th}}$  slack variable.

### Convergence

Lemma 1  $\theta_i$ , for any variable which is a candidate blocking variable, the amount it can change before it is driven to infeasibility remains the same after a constraint switch is made.

Proof: A constraint switch can only take place if for the  $i^{\text{th}}$  slack variable in the basis,  $\hat{a}_k$  the update entering vector has a negative in the  $\hat{a}_{ik}$  element, then the  $i^{\text{th}}$  slack variable is in the set of candidate blocking variables, because for unit increase in  $\hat{a}_k$  the  $i^{\text{th}}$  slack will also increase one unit and it must remain  $\leq \bar{b}_i$  (where  $\bar{b}_i = b_i^+ - b_i^-$ ) for it to be feasible.

$$\theta_i = \frac{u_i - \hat{b}_i}{-\hat{a}_{ik}} \quad \text{for } a_i < 0 \text{ in general}$$

$$u_i = \bar{b}_i = b_i^+ - b_i^-$$

(1)

$$\theta_i = \frac{\bar{b}_i - \hat{b}_i}{-\hat{a}_{ik}} \quad \text{before switch}$$

When a constraint switch takes place we change the basis by substituting slack variables. This is accomplished by multiplying the  $i^{\text{th}}$  column by -1. We must also update the  $B^{-1}$ ,  $\hat{b}$ ,  $\hat{a}_k$  by multiplying the  $i^{\text{th}}$  row of each by -1, and the value of the slack variable becomes

$\hat{b}_i = -\hat{b}_i + \hat{b}_i$ , after updating,  $-\hat{a}_{ik} > 0$ . Therefore  $\theta_i$  is  $\frac{\hat{b}_i}{\hat{a}_{ij}}$  for  $\hat{a}_{ij} > 0$  in general. So now

$$(2) \quad \theta_i = \frac{-\hat{b}_i + \bar{b}_i}{-\hat{a}_{ik}} = \frac{\bar{b}_i - \hat{b}_i}{-\hat{a}_{ik}}$$

From equations 1 and 2 we see  $\theta$  is the same so that a switch does not have to be performed in order to determine  $\theta_i$  for a slack variable in the basis.

**Lemma II** The ratio of the  $i^{\text{th}}$  row elements of the updated  $B^{-1}$  to the  $i^{\text{th}}$  element  $\hat{a}_{ik}$  remain the same even if a constraint switch is performed on the  $i^{\text{th}}$  variable in the basis.

Proof: Again, a constraint switch can be performed if the  $i^{\text{th}}$  slack variable is in the basis and  $\hat{a}_{ik} < 0$ .

$$(3) \quad \text{Ratio} = \frac{b_{ij}^{-1}}{\hat{a}_{ik}} \quad j=1,2,\dots,m$$

After switch

$$(4) \quad \text{Ratio} = \frac{-b_{ij}^{-1}}{-\hat{a}_{ik}} \quad j=1,2,\dots,m$$

Therefore the ratios from equations 3 and 4 are equal, and again a switch does not have to be performed in order to determine these ratios.

When degeneracy occurs, there is a possibility that the objective function will not increase during a sequence of iterations, and it is also possible that a basis may be repeated. If this happens the

simplex algorithm will not converge. Dantzig (2) has proposed a method of perturbation to avoid degeneracy. The general max flow problem becomes,

$$\begin{aligned} & \text{Max } \sum_{j=1}^m x_j \\ \text{s.t. } & \sum_{i=1}^m a_{ij} x_j + x_{i+m} = b_i + \epsilon^i \quad j=1,2,\dots,m \\ & x's \geq 0 \end{aligned}$$

It is shown that the value of the basis variables are of the form

$$\begin{aligned} \bar{b}_i(\epsilon) &= \sum_{k=1}^m b_{ik}^{-1} (b_k + \epsilon^k) \quad i=1,2,\dots,m \\ &= \bar{b}_i + b_{i1}^{-1} \epsilon + b_{i2}^{-1} \epsilon^2 + \dots + b_{im}^{-1} \epsilon^m \end{aligned}$$

where  $\bar{b}_i$  is value  $x_{j_i}$  when  $\epsilon = 0$ , and  $[b^{-1}]$  is the current basis inverse. It is also stated that the basic variable to leave the basis is determined by

$$x_s = \frac{\bar{b}_i(\epsilon)}{a_{is}} = \min_{a_{is} > 0} \{ (\bar{b}_i + b_{i1}^{-1} \epsilon + b_{i2}^{-1} \epsilon^2 + \dots + b_{im}^{-1} \epsilon^m) / a_{is} \}$$

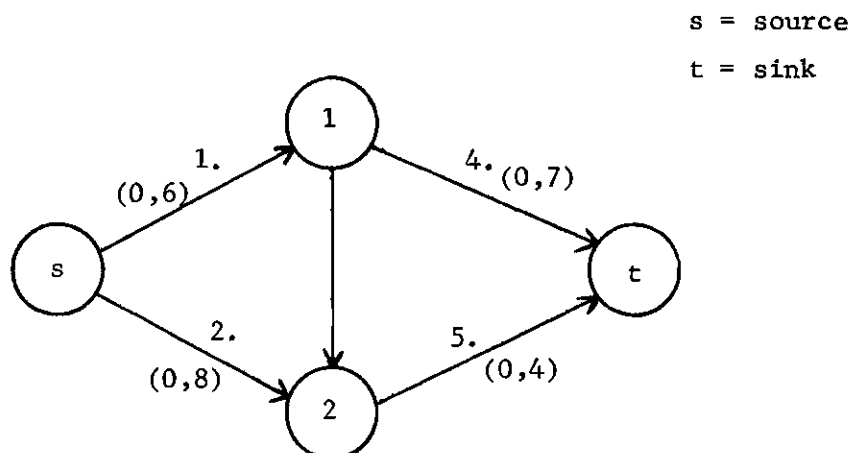
From Lemma II in Dantzig (2), the minimum of polynomial expressions is determined by first comparing the constant terms and choosing the minimum, if there is more than tied, compare the terms involving  $\epsilon$  and so forth until a unique minimum is found. We have shown by lemmas I and II that the ratios of these terms of the polynomial expressions re-

main unchanged after a constraint switch. Therefore the variable to leave the basis can be determined by

$$x_s^* = \frac{\bar{b}_r(\epsilon)}{a_{rs}} = \min_{a_{is}} \{ (\bar{b}_i + b_{il}^{-1} + \dots + b_{im}^{-1}) a_{is} \}$$

where  $\hat{a}_{is} > 0$  for path variables and  $\hat{a}_{is}$  is unrestricted for slack variables without switching any constraints. Once the leaving variable has been determined and it is a slack then a constraint switch is performed, followed by the usual simplex pivot operation. Dantzig (2) has also shown that the simplex algorithm applied to the perturbed problem will converge to an optimal value in a finite number of iterations, and that the optimal basic feasible solution to the perturbed problem is also the optimal solution to the unperturbed problem, when  $\epsilon = 0$ . Therefore we can conclude that the constraint switching procedure with the associated pivot operation will converge to an optimal value in a finite number of iterations.

#### EXAMPLE OF THE CONSTRAINT SWITCH PROCEDURE



$(b^-, b^+) =$  lower and upper bounds on the flow in the arc.

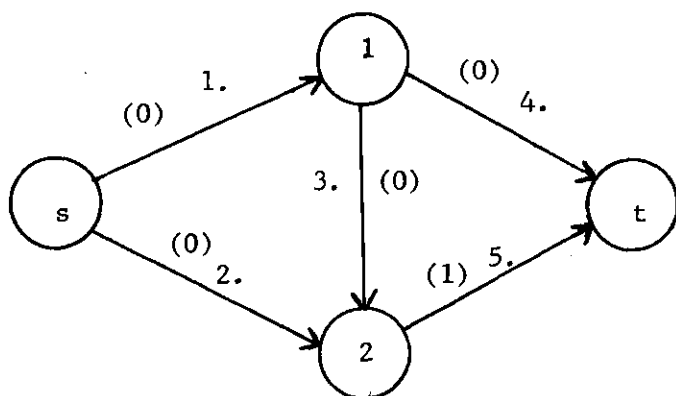
Initial basic feasible solution contains path,  $P_1 = \{1, 3, 5\}$  and upper constraint slack variables for arcs 1, 2, 3, 4, in the basis.

INITIAL BASIS						R.H.S.	
$s_1$	1	0	0	0	1	6	All constraints are upper-bounded.
$s_2$	0	1	0	0	0	8	
$s_3$	0	0	1	0	1	5	
$s_4$	0	0	0	1	0	7	
$x_1$	0	0	0	0	1	4	
$C_B$	0	0	0	0	1		

INITIAL BASIS INVERSE						$B^{-1}b$
$s_1$	1	0	0	0	-1	2
$s_2$	0	1	0	0	0	8
$s_3$	0	0	1	0	-1	1
$s_4$	0	0	1	1	0	7
$x_1$	0	0	0	0	1	4
$C_B B^{-1}$	0	0	0	0	1	Flow = 4

Shortest path column generation

$$\begin{aligned}
 \min_j z_j - c_j &= \min_j z_j \\
 &= \min_j c_B B^{-1} a_j \\
 &= \min_{P_j} \sum_{i \in P_j} \pi_i
 \end{aligned}$$



$(\pi)$  is applied to each arc

shortest path:  $P_2 = \{1,4\}$

$$z_j - c_j = (0+0) - 1 = -1$$

Enter  $x_2$

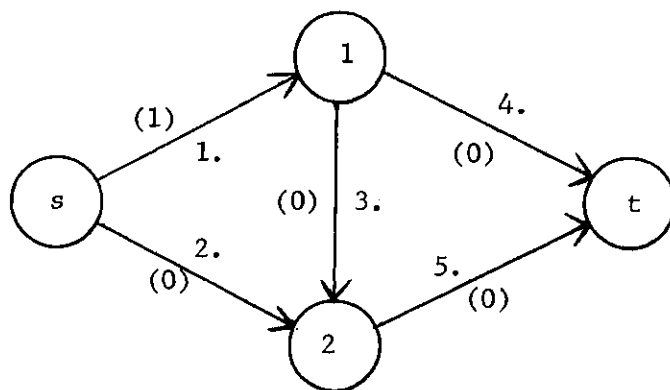
	$B^{-1}$	$\hat{b}$	$\hat{a}_2$	Updated entering vector
$s_1$	1 0 0 0 -1	2	1	
$s_2$	0 1 0 0 1	8	0	
$s_3$	0 0 1 0 -1	1	0	
$s_4$	0 0 0 1 0	7	1	
$x_1$	0 0 0 0 1	4	0	

Blocking variable is  $s_1$

	$B^{-1}$	$\hat{b}$	Flow = 6
$x_2$	1 0 0 0 -1	2	
$s_2$	0 1 0 0 0	8	
$s_3$	0 0 1 0 -1	1	
$s_4$	-1 0 0 1 1	5	
$x_1$	0 0 0 0 1	4	



$$c_B B^{-1} \quad \boxed{1 \ 0 \ 0 \ 0 \ 0}$$



Shortest Path:  $P_3 = \{2, 5\}$

$$z_j - c_j = (0+0) - 1 = -1$$

Enter  $x_3$ .

	$B^{-1}$	$\hat{b}$	$\hat{a}_3$
$x_2$	$\begin{bmatrix} 1 & 0 & 0 & 0 & -1 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & -1 \\ -1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 2 \\ 8 \\ 1 \\ 5 \\ 4 \end{bmatrix}$	$\begin{bmatrix} -1 \\ 1 \\ -1 \\ 1 \\ 1 \end{bmatrix}$
$s_2$			
$s_3$			
$s_4$			
$x_1$			

Candidates for blocking variable  $s_2, s_3, s_4, x_1$

- 1)  $x_3' = x_3 + \theta = \theta$  ( $\geq 0$ )
- 2)  $s_2' = s_2 - \theta = 8 - \theta$  ( $\geq 0$ )
- 3)  $s_3' = s_3 + \theta = 1 + \theta$  ( $\leq 5 - 2$ )
- 4)  $s_4' = s_4 - \theta = 5 - \theta$  ( $\geq 0$ )
- 5)  $x_1' = x_1 - \theta = 4 - \theta$  ( $\geq 0$ )

- 1)  $\theta \geq 0$

- 2)  $\theta \leq 8$

- 3)  $\theta \leq \frac{3-1}{1} = 2$

- 4)  $\theta \leq 5$

- 5)  $\theta \leq 4$

$$\theta = \min_i \left\{ \frac{b_i}{a_{ij}} \mid a_{ij} > 0 \right\} = 4$$

$$\theta = \min_i \left\{ \frac{b_i^+ - b_i^-}{-a_{ij}} \mid a_{ij} < 0 \right\} = 2$$

$$\theta_3 = u_3 = \infty$$

$$\theta = \text{blocking variable} = 2 \rightarrow s_3$$

The lower constraint for arc 3 is blocking. Therefore exchange the lower constraint for the upper constraint, and then pivot.

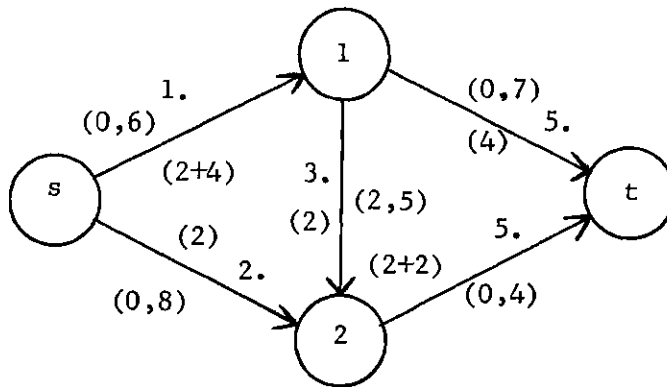
	$B^{-1}$	$\hat{b}$	$\hat{a}_3$
$x_2$	1 0 0 0 -1	2	-1
$s_2$	0 1 0 0 0	8	1
$t_3$	0 0 -1 0 1	2	1
$s_4$	-1 0 0 0 1	5	1
$x_1$	0 0 0 0 1	4	1

$$-\hat{a}_{ik} = (-B_i^{-1})a_k$$

$$\hat{b}'_{ik} = -\hat{b}_3 + (b_3^+ - b_3^-) = -1 + (5-2) = 2$$

<u>TYPE</u>		$B^{-1}$	$\hat{b}$
+	$x_2$	1 0 -1 0 0	4
+	$s_2$	0 1 1 0 -1	6
-	$x_3$	0 0 -1 0 1	2
+	$s_4$	-1 0 1 1 0	3
+	$x_1$	0 0 1 0 0	2

$$C_B B^{-1} \quad \begin{bmatrix} 0 & 0 & -1 & 0 & 1 \end{bmatrix}$$

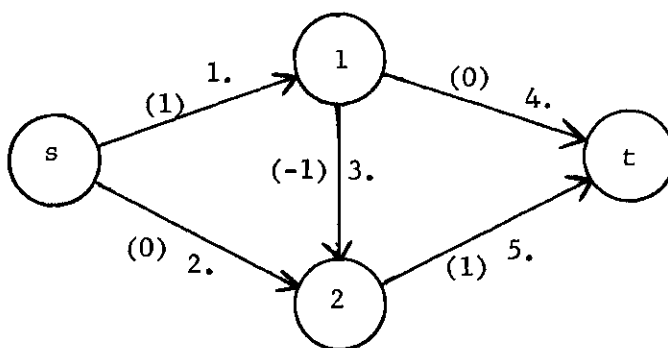
OPTIMAL SOLUTION

max flow = 8

$$P_1 = \{1, 3, 5\} \quad x_1 = 2$$

$$P_2 = \{1, 4\} \quad x_2 = 4$$

$$P_3 = \{2, 5\} \quad x_3 = 2$$



Shortest path:  $\{1, 3, 5\}$  has length = 1. Therefore, the above solution is optimal.

CHAPTER VI

THE CONSTRAINED SHORTEST PATH ALGORITHM

FOR ACYCLIC NETWORK

In a shortest path network problem, there are lengths associated with the arcs of the network. These lengths can represent time, cost or distance. The shortest path problem determines the minimum length path from the source node to the sink node through the network. The constrained shortest path problem has been formulated by Berry and Jensen (6) such that in addition to a length on each arc, there is also a requirement of some limited resource. So the constrained shortest path problem becomes one of determining the minimum length path through the network subject to these resource restrictions. Hinkle (1) has shown that a constrained shortest path problem can be used to generate a path through the network of maximal flow problem, which will possibly reduce the flow on the longest path and yet retain maximal flow.

The mathematical formulation of Hinkle's constrained shortest path problem is

$$\begin{aligned}
 (1) \quad & \text{Min} \quad (p \pi - \sum_{j \in L} B^{-1}) a_j \\
 & \quad V_j, \text{ nonbasic} \\
 \text{s. t.} \quad & L_j \leq L^* - 1
 \end{aligned}$$

The vector  $a_j$ , which does in fact minimize this objective function will be a path through the network which will: (1) insure that the maximal flow through the network is maintained; (2) tend to reduce

the flow on the path or set of paths with the longest length through the network; and (3) have a length less than the longest path. Since the constrained shortest path problem is searching for that returns an optimal value to problem (1), let us reformulate the problem in terms of the set of all paths through the network. In general terms, the constrained shortest path problems may be stated as: determine the path through the network which has the minimal sum of costs on the arcs which make up the path, and has a sum of lengths on the arcs which is less than or equal to the length of the longest path minus one. Now the problem of finding enter path is defined in terms of the arc costs and lengths of the network, and can be formulated as,

$$\begin{aligned}
 (2) \quad & \text{Min} \quad \sum_{p \in P} c_{ij} x_{ij} \\
 \text{s. t.} \quad & \sum_{i,j \in P} l_{ij} x_{ij} \leq L^* - 1 \\
 & x_{ij} = 0, 1
 \end{aligned}$$

$$\text{where } c_{ij} = \text{cost of using arc } (i, j) = p\pi_i - \sum_{\ell=1}^m B_{\ell}^{-1} l_{\ell i}$$

$$l_{ij} = \text{length of arc } (i, j)$$

$L^* =$  length of the longest path(s) in the network carrying flow

$$x_{ij} = \begin{cases} 1, & \text{if arc } (i,j) \text{ is used in path } p \\ 0, & \text{otherwise} \end{cases}$$

$P =$  set of all paths,  $p$  through the network

Problem (2) can be solved by the arc costs,  $c_{ij}$ , and the arc

lengths,  $\ell_{ij}$ , to the network and determining the minimal cost path through the network such that its length is less than or equal to  $L^* - 1$ . Since there exists a possibility of negative arc costs if the constrained shortest problem is used to generate a "reducing" path for the ship scheduling problem, the solution procedure must take advantage of the acyclic property of the network representing the ship scheduling problem if negative cost directed cycles are to be avoided.

#### The Constrained Shortest Path Algorithm

The following labelling procedure is a modification of Hinkle's procedure in that the network must be acyclic and negative arc costs are allowed.

#### ALGORITHM:

$x$  is defined as a scanned set, which includes a node that has been labeled, or a node for which it has been determined it could not be labelled.

$A$  is the arc set for the network.

STEP 0: Label the sink,  $t$ , with

$$L_1(t) = (0, 0, \infty) \text{ and place } t \text{ in } x$$

STEP 1: Find a node  $i$  which can be labeled: a node,  $i$ , such that

$$(i, j) \in A \rightarrow j \in x.$$

There must be at least one if  $s$ , the source, is not in  $x$ .

STEP 2: If A node,  $i$ , is found construct

$$L_k(i) = (\ell(j) + \ell_{ij}, c(j) + c_{ij}, (i, j))$$

where  $l_{ij}$  = length of arc (i,j)

$c_{ij}$  = cost of arc (i,j)

$l(j)$  = length to node j from sink

$c(j)$  = cost to node j from sink

If any two labels have the same length take the one with the smaller cost.

Place i in x.

STEP 3: Repeat Steps 1 and 2 until the source, s, has been labeled, or until it has been determined that s cannot be labeled because the length from the source to the sink (the shortest path) is greater than  $L^* - 1$ .

Rule 1:

Do not construct labels for node i until all nodes j, such that  $(i, j) \in A$ , are elements of set x (i.e., all nodes j have been labeled).

Remark 1:

Since we are searching for the minimum cost, shortest path from s to t, we wish to construct a label list for node i which contains only minimum length and cost labels. So the resulting list for i is a subset of all possible labels that could be generated by the above procedure and Rule 1. If the second part of Step 2 is not employed we will generate labels which do not indicate minimum length and cost, partial paths from node i to the sink.

Remark 2:

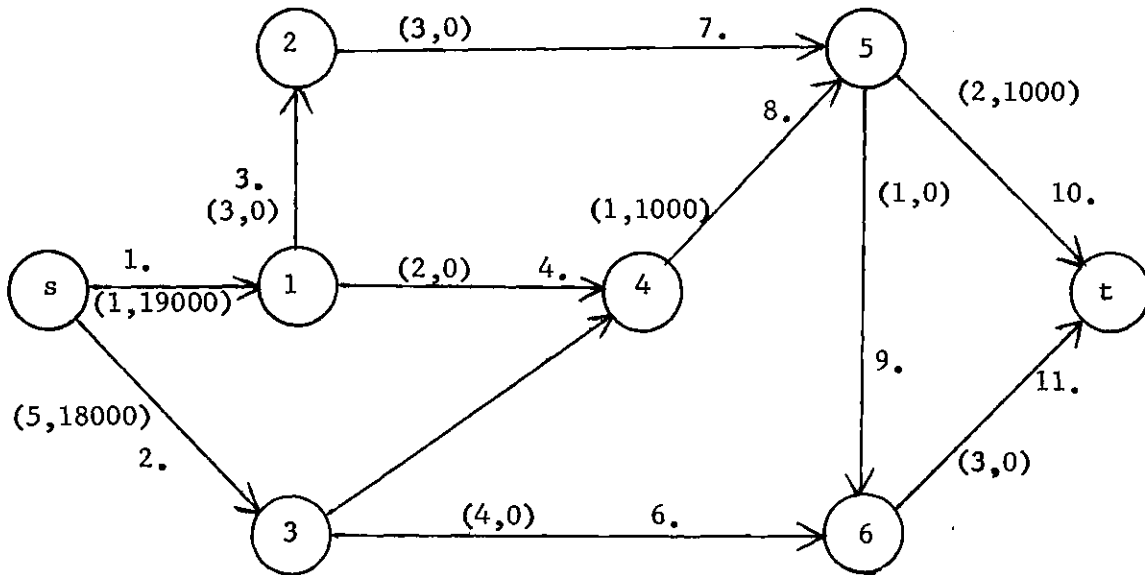
If at any stage of the labeling procedure the length of all k



labels from node  $j$   $\ell(j) \forall k \geq L^* - 1$  and  $j \neq s$  then this node  $j$  and all arcs incident to node  $j$  can be removed from the network. This removal operation will avoid unnecessary calculations in the labeling procedure, because the path generated using this node will not meet the requirements to be a candidate to enter basis, as defined by Hinkle (3), i.e., a  $P_j$   $L^*-1$ . A simple way to find a node which can be labeled is to pick any node,  $i$ , not in  $x$ . Then if all the successors,  $j$ , to node  $i$  (i.e., those  $j$  for which  $(i,j) \in A$ ) are labeled then node  $i$  can be labeled. Otherwise, pick some successor,  $j$ , not in  $x$  and try to label it. Eventually some node must be found which can be labeled since  $t$  is in  $x$  and the network is assured to be acyclic.

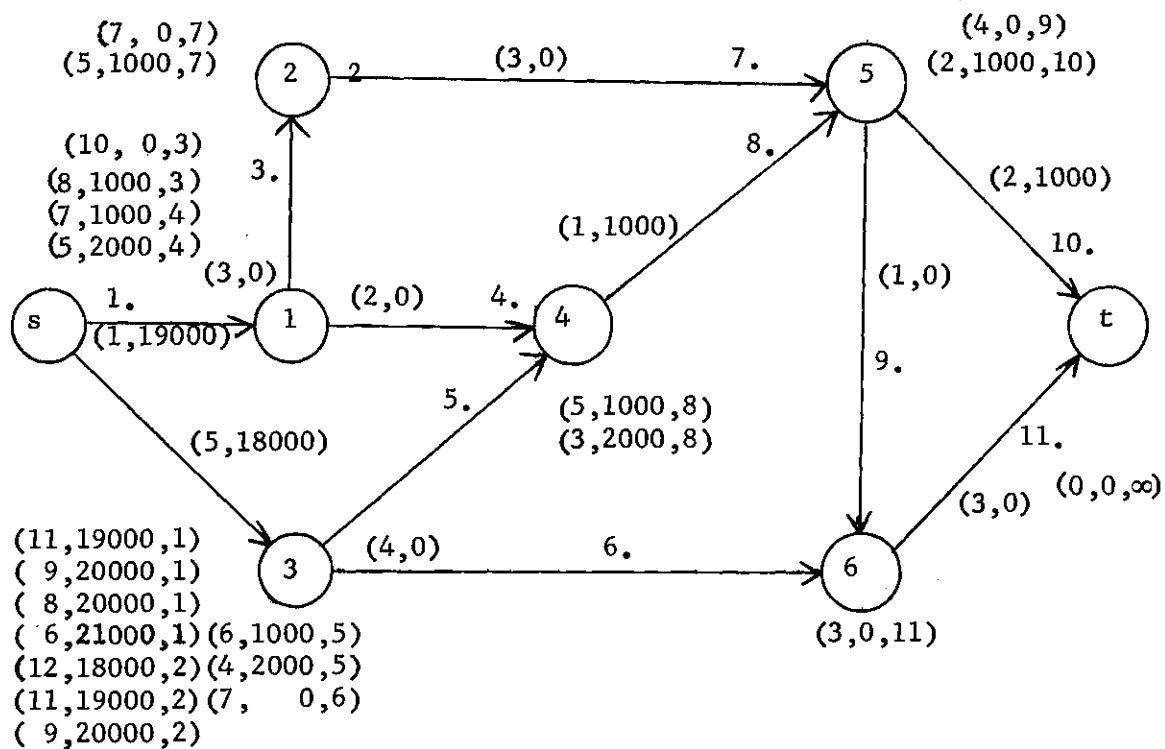
At each iteration of the algorithm a new node receives all of its labels. Thus, the algorithm requires exactly as many iterations as there are nodes in the network.

An Example of the Constrained Shortest Path Procedure



$(l_{ij}, c_{ij})$  = Length and cost on  $(i,j)$  A from Hinkle's (1) example  
 with the costs being generated from  $p\pi - \sum_{k=1}^L B_k^{-1}$  for  
 the last iteration of .CSPATH in the computer program.

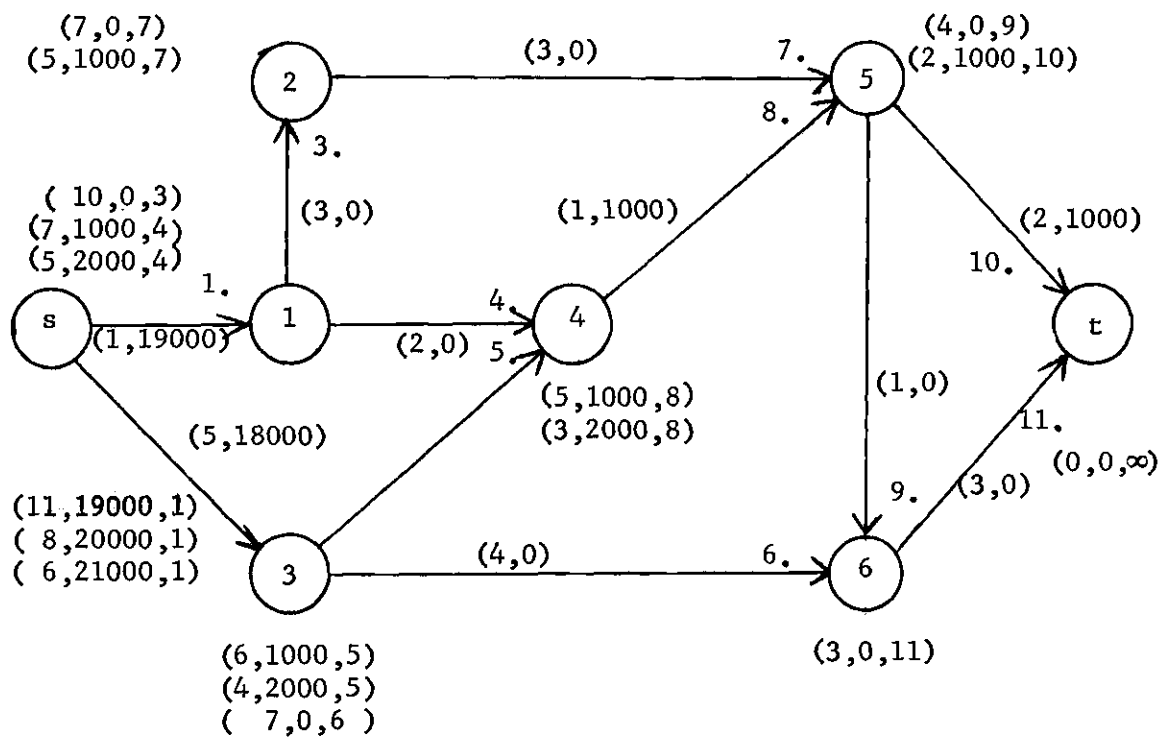
Network Below has All Possible Labels Shown



LABEL = (LENGTH, COST, ARC NOS.)

LSTAR =  $L_B - 1 = 12 - 1 = 11$

Network Below Shows Final Labels



Label = (Length, Cost, Arc Nos. )

Label at s, that generates candidate path is  $(8, 20000, 1)$  and the path is 1-4-8-9-11.

## CHAPTER VII

## EXAMPLE

STEP 1 SOLUTION OF THE SUPPLY TRANSPORTATION SUBPROBLEM

I) GIVEN

 $a_i$ 

A) Available load equivalents  
of supplies at the bases

1
4

B) "Base to Port" travel time matrix, BP

		PORTS	
		1	2
B A S E S	1	6	2
	2	3	4

C) "Ship to Port" travel time matrix, SP

		PORTS	
		1	2
S H I P S	1	6	2
	2	3	1
	3	1	5
	4	4	3
	5	3	5

D) "Port to Objective" travel time vector, PO

OBJECTIVE

P O R T S	1	4
	2	3

The required load equivalents (ship loads) of supplies at the ports were determined as;  $b_i$  thus the number of ships that must

2
3

travel to port 1 is 2 and to port 2 is 3.

## II) SOLUTION OF THE LEAST TIME TRANSPORTATION PROBLEM

		PORTS		
		1	2	$a_i$
BASIS	1	6	2	1
		1		
	2	3	4	4
		1	3	
$b_j$		2	3	5

$$\begin{aligned}
 T_1 &= \max \{t_{11}, t_{21}, t_{22}\} \\
 &= \max \{6, 3, 4\} \\
 &= 6
 \end{aligned}$$

		PORTS		
		1	2	
BASIS	1	6	2	1
			1	
	2	3	4	4
		2	2	
		2	3	5

OPTIMAL SOLUTION:

$$\begin{aligned}
 x_{12} &= 1 & t_{12} &= 2 \\
 x_{21} &= 2 & t_{21} &= 3 \\
 x_{22} &= 2 & t_{22} &= 4 \\
 z &= t_{12} + t_{21} + t_{22} = 9
 \end{aligned}$$

CLOSURE TIME = MAX  $t_{12}, t_{21}, t_{22} = 4$  (i.e. all the supplies will have arrived at the ports in 4 time units).

### III) NETWORK CONSTRUCTION

The NETWORK was constructed according to the algorithm developed in Chapter 3, and it appears in Figure 4. A renumber version of this network appears in Figure 5.

### IV) APPLICATION OF UPPER AND LOWER CAPACITIES, AND LENGTHS TO THE ARCS

The first network shown in Figure 4, was constructed and labelled using the procedures described in Chapter 3. However, the computer program will actually construct the ship scheduling network internally. The second network shown in Figure 5, is the renumbered version of the network in Figure 4. In comparison, the network in Figure 5 has arcs (21, 2t), (11, 1t) and (12, 1t) removed since no flow can move through them due to the upper and lower arc capacities equalling zero. Nodes 10, 16, and 17 have been added to the network, to prevent any excess units of flow from moving through the port portions of the network (i.e. node 16 was added to prevent more than one ship from leaving port one before time four, when the next units of supplies arrive at port one). The node and arc numbers on this network correspond to those numbers generated internally by the computer program.

### STEP 2 SOLUTION OF THE SHIP SCHEDULING SUBPROBLEM

#### OPTIMAL SOLUTION

<u>BASIS</u>	<u>FLOW</u>	<u>LENGTH</u>	<u>PATH</u>
2	.00	0	1
3	.00	0	2

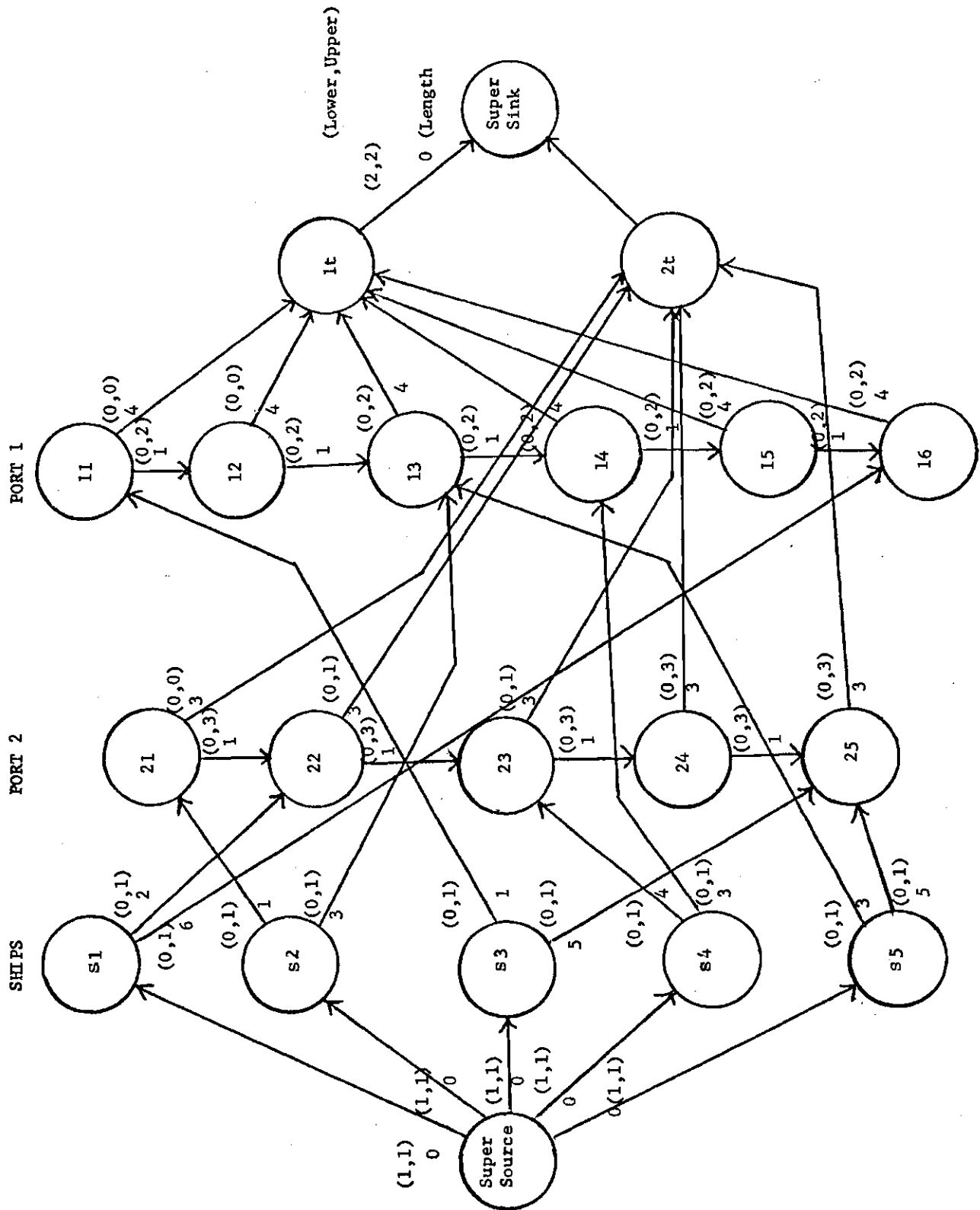


Figure 4. The Ship Scheduling Network for the Example.



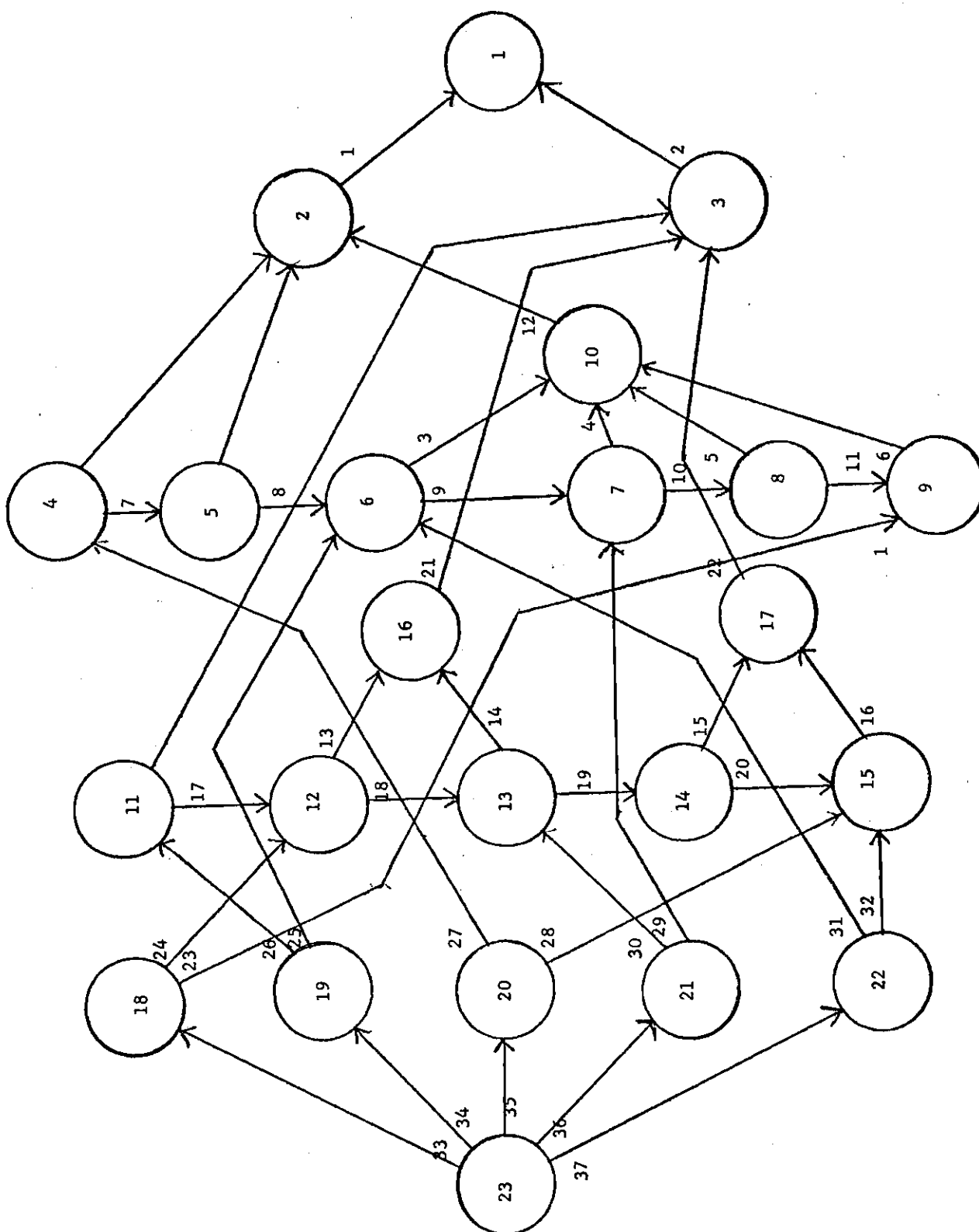


Figure 5. The Ship Scheduling Network The Example Renumbered.

## OPTIMAL SOLUTION (Continued)

<u>BASIS</u>	<u>FLOW</u>	<u>LENGTH</u>	<u>PATH</u>
4	.00	0	3
5	2.00	0	4
6	2.00	0	5
7	2.00	0	6
8	1.00	0	7
9	1.00	0	8
10	2.00	0	9
11	2.00	0	10
12	2.00	0	11
13	.00	0	12
14	.00	5	34 26 17 13 21 2
15	1.00	0	14
16	1.00	0	15
17	3.00	0	16
18	2.00	0	17
19	2.00	0	18
20	1.00	0	19
21	3.00	0	20
22	.00	6	33 24 18 14 21 2
23	1.00	0	22
24	1.00	0	23
25	.00	0	24
26	1.00	0	25
27	.00	0	26
28	.00	0	27
29	1.00	0	28
30	1.00	0	29
31	.00	0	30
32	.00	0	31
33	1.00	0	32
34	1.00	5	33 24 13 21 2
35	1.00	7	34 26 17 18 19 15 22 2
36	1.00	7	35 27 7 8 3 12 1
37	1.00	7	36 30 19 15 22 2
38	1.00	7	37 31 3 12 1
END	3169 MLSEC		

The above listing is the optimal solution to the ship scheduling subproblem for this example. This solution must now be interpreted to give the schedule of the ships to the ports. Only paths with positive flow and length correspond to actual routings of ships through the network. Each path contains the arcs of the second network that are

used to make up the path. Thus the schedule of the ships may be obtained by comparing these paths with the input networks. The first path with positive flow and length is the path using arcs 33, 24, 13, 21, 2. This path corresponds to assigning ship 1 to port 2 and it will arrive, at port 2 at time 2 and the objective area at time 5. The complete solution can be interpreted as above and is shown in the following table.

SHIP	PORT	ARRIVAL TIME	
		AT THE PORT	AT THE OBJECTIVE
1	2	2	5
2	2	1	7
3	1	1	7
4	2	3	7
5	1	3	7

Ships 1 and 2 arrive at each port at time 1 but they must wait 3 and 2 time periods respectively, until supplies arrive at the ports before they can take on these goods and travel to the objective area.

## CHAPTER VIII

## PROCEDURES FOR

## COUPLING THE TWO SUBPROBLEMS TOGETHER

The least time transportation problem, in Chapter II, requires explicit knowledge of the quantities of supplies that must eventually arrive in the ports (i.e., the  $b_j$  values must be known). These requirements for supplies become the linking mechanisms between the supply transportation and ship scheduling subproblems, since each ship load of supplies that must be transported from a base to a port implies that a ship must travel to that port to take on these supplies. Once the port requirements are determined the number of ships that must be scheduled through each port is known, however, the actual assignment of ship  $i$  to some port  $j$  is determined by the second subproblem. Since the number of ship loads of supplies available at each inland base, and the number of ships necessary to transport these are known inputs to the master problem, the only variables are the port requirements. Having selected a set of  $b_j$  values the schedule of ships generated by the second subproblem is only optimal (minimal closure) with respect to these port requirements. Thus, if the  $b_j$ 's were altered in some manner, an improved schedule could be obtained. A branch and bounding scheme would seem to have application to this sort of problem.

### Branch and Bound

The strategic transportation problem may now be restated as: determine the set of port requirements for the Bases-to-Ports problem that will generate a ship schedule in the Ships-to-Ports problem, which has an overall minimal closure time. This problem would fall into the class of problem, known as combinatorial problems. Agin (7) has defined a combinatorial problem to be one of assigning discrete numerical values to some finite set of variables  $X$ , in such a way as to minimize an objective function subject to a set of constraints. Let us relate this definition to our problem. We would like to assign a requirement value to each port such that the sum of the assigned values equals the number of ship loads of supplies available at the bases and generates a minimal closure schedule of the ships (i.e. minimizes the objective function of closure at the objective area). Branch and bound algorithms can be constructed to solve these types of problems, and they may be viewed as creating a tree consisting of nodes and branches between the nodes. The collection of all solutions to the combinatorial problem can be thought of as the root of the tree and branches indicating a partitioning process of the collection into smaller and smaller collections of the solutions with each of these collections represented by a node. A branch and bound algorithm is defined by Agin to be a set of rules for (1) branching from nodes to new nodes, (2) determining lower bounds for new nodes, (3) choosing an intermediate node from which to branch next, (4) recognizing when a node contains only infeasible or non-optimal solutions and (5) recognizing when a

final node contains an optimal solution.

If there did not exist an initial set of port requirements from which a search for the best set could be started from, the tree created by a branch and bound algorithm would have a root containing all possible sets of port requirements. Let us now define a process that would partition this collection of all possible solutions into simpler collections of solutions. Let the partitioning consist of assigning a requirement value to a particular port and allowing the remainder of the available supplies to be assigned to the other ports in all the ways possible with the restriction the sum of all the assigned requirement values must equal the number of available units of supplies. If the total units of supplies available at the bases is five and there are three ports to which the supplies can be assigned, one node of the tree would represent the sets of port requirements such that there was one unit assigned to the first port and the other four units were assigned in all combinations to the other two ports. Since we are seeking the set of port requirements which renders an overall minimal closure ship schedule, let us define a lower bound for each node of the tree to be the minimal closure time of the schedules that could be generated from the set of port requirements represented by the particular node. A new node would be selected based on the minimal value of the bounds at the same level of the tree. The following figure shows a portion of the tree that would be created for a three port and five units of supplies problem. The bounds shown do not represent closure values from an actual ship schedule, but are simply used to indicate

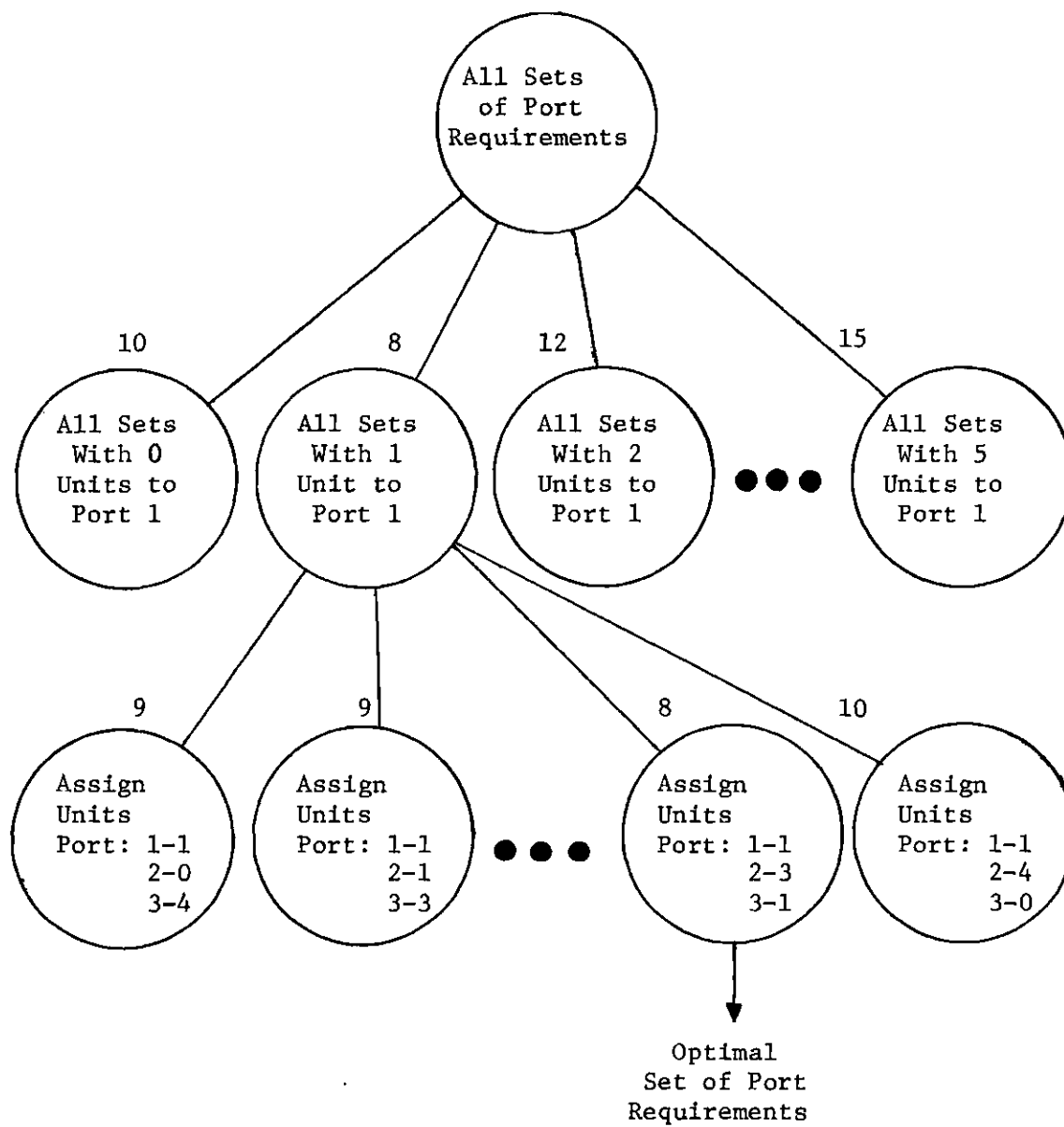


Figure 6. A Portion of a Tree Created by a Branch and Bound Algorithm.

how the branching would take place. With the bounds used the optimal set of port requirements would be one unit at port one, two units at port two and two units at port three.

Given an initial set of port requirements, a closure time for the master problem could be obtained through the solution of both sub-problems and this time could be used as a bound on that particular solution (i.e., assignment of supplies to ports and ships to the ports). Then a branching scheme could be invoked to change certain of the port requirements and this new master problem could be solved. A continued branching and bounding could be employed until an overall minimal closure time solution was generated. Each solution in the branch and bound method would be the specification of requirements at the ports for the Bases-to-Ports transportation problem. Branching would result in a new assignment of requirements which would either lead to improvement in the overall closure time and a better solution or give indications that other branchings must be made.

A branch and bound procedure which enumerates requirements at the ports could be cumbersome and (algorithmic) time consuming. Methods need to be found which could accelerate the branch and bound procedure and/or which could lead to good heuristic solutions to the total problems.

#### Utilizing Linear Programming Information in the Branch and Bound Procedure

In the previous section a branch and bound procedure was discussed which combined the Bases-to-Ports least time transportation



problem with the Ships-to-Ports min-max flow problem. In review, lower and upper bounds were placed on certain arcs of the Ships-to-Ports problem in order to "force" ships through specified ports and specific times. These bounds would be generated from the solution to the Bases-to-Ports problem.

In this section we shall describe methods by which the solution information to the Ships-to-Ports problem can be used to generate a new solution to the Bases-to-Ports problem. This would close the loop in the branch and bound process and thereby, produce an iterative method for solution to the total problem. We begin by presenting some basic economic results from linear programming theory.

The basic linear programming problem can be stated in matrix form as:

$$\begin{aligned} \text{Min } z &= CX \\ \text{s. t. } AX &= b \\ x &\geq 0. \end{aligned} \tag{1}$$

Suppose we know that  $B \leq A$  is an optimal basis. Then partitioning

$A = (B, N)$ ,  $C = (C_B, C_N)$ ,  $X^* = (X_B^*, X_N^*)$  we get

$$z^* = C_B X_B^* + C_N X_N^*$$

and

$$BX_B^* + NX_N^* = b.$$

$$X_B^* = B^{-1}b + B^{-1}NX_N^*$$

and the optimal basic feasible solution is  $X_B^* = B^{-1}b$ ,  $X_N^* = 0$ ,  $z^* = C_B B^{-1}b$ . Let us define

$$\pi_B = C_B B^{-1}.$$

These are the dual variables associated with the optimal solution to the above l.p. problem. Then  $z^* = \pi_B b$ ,

Assuming regularity of the optimal basis under small perturbations of the  $b$ -vector we can use  $z^* = \pi_B b$  to obtain

$$\frac{\partial z^*}{\partial b} = \pi_B$$

or

$$\frac{\partial z^*}{\partial b_i} = \pi_{B_i}. \quad (2)$$

This has a very powerful economic interpretation. In effect, equation (2) indicates that if the values of some  $b_i$  is changed by a small amount  $\epsilon$  (and the problem were resolved) then we would expect the optimal objective value,  $z^*$ , to change by an amount  $\pi_{B_i} \cdot \epsilon$ . This provides us with information as to how the right hand side values,  $b_i$ , might be changed so as to improve the optimal value of the objective function.

Again, assuming a regularity condition on the optimal basis under

small perturbations in the  $b$ -vector, we can use  $X_B^* = B^{-1}b$  to obtain

$$\frac{\partial X_B^*}{\partial b} = B^{-1}$$

or

$$\frac{\partial X_{B_k}^*}{\partial b_i} = B_{ki}^{-1} \quad . \quad (3)$$

Interpreting (3) we find that a small change  $\epsilon$  in some  $b_i$  is expected to produce a change in  $X_{B_k}^*$  in the amount of  $B_{ki}^{-1} \cdot \epsilon$ . As before, this indicates which  $b_i$  might be changed in order to offset desired changes in the optimal values of the variables. We shall utilize (3) to "drive" the min-max flow problem to a better solution.

Recall, in the Ships-to-Ports problem we are trying to minimize the closure time, i.e., the length of the longest path carrying flow. For fixed lower and upper bounds we utilized linear programming and the constrained shortest path problem to successively alternate optimal basic feasible solutions in which the flow on the longest path (with positive flow) is being driven to zero. When that algorithm stopped an optimal solution (one in which the longest path carrying flow was as short as possible) was obtained for the prescribed set of lower and upper bounds on arc flow. It is important to note that these lower and upper bounds were used as right hand side values,  $b_i$ , in the linear programming problem. Therefore, when the

algorithm finally stops we could apply result (3) to determine what effect changes in each  $b_i$  (each lower and/or upper bounds) is expected to have on the flow in the longest path. Specifically, we could determine which  $b_i$ 's (which lower and upper bounds) could be changed, and in what direction, so that the flow in the longest path might be moved toward zero. This would in turn indicate which supplies have to be changed, either in arrival time or destination, at the ports.

As an illustration of how this method can be utilized consider the example problem of Chapter IV. Information associated with the optimal solution for that problem is:

<u>TYPE</u>		$B^{-1}$	$b$
+	$x_2^*$	1 0 -1 0 0	4
+	$s_2^*$	0 1 1 0 -1	6
-	$x_3^*$	0 0 -1 0 1	2
+	$s_4^*$	-1 0 1 1 0	3
+	$x_1^*$	0 0 1 0 0	2

$$C_B B^{-1} \quad 1 \quad 0 \quad -1 \quad 0 \quad 1$$

$$P_1 = \{1, 3, 5\}$$

$$P_2 = \{1, 4\}$$

$$P_3 = \{2, 5\}$$

The example did not have lengths specified for arcs, but, this is not important to our discussion here.

Suppose path 2 had been the longest path in the optimal solution (above). Considering the first row of  $B^{-1}$  we find that

$$\frac{\partial x_2^*}{\partial b_1} = 1, \quad \frac{\partial x_2^*}{\partial b_3} = -1, \quad \frac{\partial x_2^*}{\partial b_i} = 0 \quad i \neq 1 \text{ or } 3.$$

This indicates that there are potentially two ways to lower the flow,  $x_2^*$ , on path 2. One way is to lower the right hand side value associated with constraint #1. From the TYPE indication we see that the upper constraint was active for arc #1. Hence, one way to reduce the flow,  $x_2^*$ , on path 2 is to decrease the upper capacity,  $b_1^+$ , on arc #1. This is reasonable since  $x_2^* = 4$ ,  $b_1^+ = 6$ , and the additional two units of capacity are being utilized by path 1 ( $x_1^* = 2$ ).

A second way to lower the flow on path 2 is to raise the right hand side value associated with constraint #3. From the TYPE indication we see that the lower constraint was active for arc #3. Therefore to lower the flow on path 2 we should raise the lower capacity,  $b_3^-$ , on arc #3. Again this is reasonable. If  $b_3^-$  is increased more flow is forced onto path 1 since path 1 is the only path using arc #3. However, since path 1 also shares arc #1 with path 2, any increase in flow on path 1 must be accompanied by a corresponding decrease in flow on path 2 as arc #1 is saturated.

In a like manner the other entries in  $B^{-1}$  can be interpreted as effecting changes in flow (or slacks).

### Total Travel Time Utilized in a Hueristic Solution Procedure

In the last section an iterative method for solution of the total strategic transportation problem was discussed. It consisted of a branch and bound procedure which allows the solution information obtained from the Ships-to-Ports problem to generate a new solution to the Bases-to-Ports problem. This iterative method, utilizing the linear programming information available from the Ships-to-Ports problem solution to generate a new set of requirements at the ports, assures an optimal solution to the total strategic transportation problem will be found, if a backtrack scheme is included in the branch and bound process. In this section a hueristic method, utilizing the total travel time of all of the ships will be discussed. Generally this method will use the solution information from the Bases-to-Ports problem coupled with the travel time information of the ships to generate a new requirement set at the ports. This requirement set will be successively altered until an optimal set in terms of a total travel time criterion is found. The Ships-to-Ports problem is then solved using this optimal requirement set. The schedule of the ships to the ports generated, will be a near optimal schedule for the total strategic transportation problem.

The travel of a ship in the Ships-to-Ports subproblem can be represented by the time units required for the prescribed travel rather than distances traveled. The total travel time of a particular ship is defined to be the sum of (1) the travel time of the ship from its location at sea to the scheduled port, (2) any delay experienced by the

ship at the port due to waiting for samples which have not arrived at the port, and (3) the travel time of the ship from the port to the objective area. The total ship travel time can be expressed in equation form as,

$$T_{ij}^j = SP_{ij} + d_i^j + PO_j, \text{ for ship } i \text{ scheduled to port } j$$

where  $SP_{ij}$  is the  $ij^{\text{th}}$  element of the Ship-to-Port travel time matrix,

$d_i^j$  is the delay experienced by ship  $i$  at port  $j$

$$d_i^j = \max \{ 0, SP_{ij} - t_{ij} \}$$

$t_{ij}$  is the arrival time of the  $i^{\text{th}}$  ship load of supplies at port  $j$ ,

and  $PO_j$  is the  $j^{\text{th}}$  element of the port-to-objective area travel time vector.

If all the ships were truly located randomly at sea, then there would be a probability associated with each location and the total ship travel time  $T_i^j$ , would then become an expected value of travel time. The SP matrix values would have to be weighted with the probability of the ship being at that particular location. The sum over all possible locations of the ship would become the  $SP_{ij}$  portion of  $T_i^j$ . Since the exact location of each ship is known with certainty (i.e., with a probability of one), the value of total travel time for ship  $i$ ,  $T_i^j$  is deterministic (since PO vector is also known with certainty). Since each ship can be scheduled to one and only one port, let  $\delta_i^j$  be an indicator variable such that,

$$\delta_i^j = \begin{cases} 1, & \text{if ship } i \text{ is scheduled to travel to port } j \\ 0, & \text{otherwise} \end{cases}$$

Now the total travel time for all ships in the Ships-to-Ports sub-problem will be

$$\sum_{i \in S} \sum_{j \in P} \delta_i^j T_i^j \quad \text{where } S \text{ is the set of all ships,} \\ \text{and } P \text{ is the set of all ports.}$$

For a given set of port requirements, the Ships-to-Ports problem will determine this minimal closure schedule of the Ships to the ports. Closure is defined as the time that the last ship carrying supplies arrives at the objective area. If the actual schedule of the ships was known, the delay component of each ship's travel time could be determined accurately and the  $T_i^j$ 's would reflect the actual travel times. And closure would be the  $\max_{i \in S} \{T_i^j\}$ . To determine the overall optimal solution of this strategic transportation problem, the set of port requirements must be found which will result in a ship schedule that has a closure time which is minimal over the set of all possible schedules. If the set of port requirements is changed, the schedule of the ships to the ports will also change. So we would like to change the port requirements in such a way, that the schedule of the ships generated by the Ships-to-Ports problem would have the minimal closure time (i.e. the minimal  $\max_{i \in S} \{T_i^j\}$ ). Alternately, if the set of port requirements was found that resulted in the set of  $T_i^j$ 's which contained the minimal travel time for each ship, then the ship schedule would



also be the minimal closure schedule. Since the Ships-to-Ports subproblem will not be explicitly solved until the best set of port requirements is determined, the delay component of the ship travel time will not be known exactly, but it will be taken as the  $\max \{0, SP_{ij} - t_{ij}\}$ . The ship travel times,  $T_i^j$ 's, will still reflect a travel time which is closed to the actual travel time values. The solution method discussed in this section will be based on determining the set of port requirements which will generate the set of minimal ship travel times, but it will not be able to guarantee that the ship schedule resulting from this set of port requirements is the overall minimal closure schedule of the strategic transportation problem. The method will however generate a near optimal solution to the strategic transportation problem. The major advantage of this method over the branch and bound procedure discussed in the previous section is that the solution of the Ships-to-Ports problem is not required at each iteration to determine how to alter the port requirements such that the overall closure time will tend to improve.

A method must be found to alter the port requirements so that the set which corresponds to the minimal values of the  $T_i^j$ 's is determined. One method might be to determine the port requirements so that the first ship has a minimal travel time, then adjust them so that the second ship has a minimal travel time and so forth, until all the ships in the set  $S$  have a minimal travel time. However, this type of sequential alteration would seem to be an inefficient process, computationally. The travel times of all the ships and how they are

effected by the particular change in the port requirements is not taken into account by this sequential process. Since there does exist an interaction between changes in the port requirements and the travel time of the ships the complete set of ship travel times must be taken into account during the alteration process. To accomplish this, let us use the total travel time of all of the ships instead of the individual travel times during the alteration scheme of the port requirements. Determining the set of port requirements that would achieve a minimal value of the total travel time for all of the ships would insure that the individual travel times would be closed to their minimal values. In general, however, minimizing the sum of numbers is not the same as minimizing the maximum and then summing the numbers (this has been shown in relation to both the least-time transportation model and the min-max path flow algorithm). Using the total travel time as a criterion for altering the port requirements will in general obtain a set of port requirements that would generate a good solution to the ship scheduling problem and in turn a good overall solution to the strategic transportation problem.

A heuristic algorithm is now presented which uses the total travel time of all ships as a criterion for altering the port requirements of the Bases-to-Ports problem in order to determine the best possible schedule of the ships to the ports. In general terms, this algorithm begins with a feasible set of port requirements, determines the total travel time for all ships, and alters the requirements in such a way as to reduce the total travel time. A judicious choice of

initial port requirements can significantly reduce the number of alteration that must occur before the best set of port requirements in terms of the total travel time is found. Since the ships are assumed to be randomly located at sea, but with a travel time matrix,  $SP$ , known with certainty, a set of port requirements which is not biased in favor of any one particular port would be a good initial choice. Hence, the algorithm is started with as even as possible distribution of the port requirements (i.e. for a given set of supply availabilities at the bases attempt to make the port requirements all equal).

#### ALGORITHM:

- (1) Obtain an initial feasible set of port requirements.
- (2) Solve the least time transportation problem discussed in Chapter II for the given set of supply availabilities and the current set of port requirements.
- (3) Assign the required number of ships to each port so that all the port requirements are met. This assignment should correspond to the best assignment as indicated by Step 5.
- (4) Calculate the total travel time of all the ships,  $\tau$ , using  $SP$  matrix,  $PO$  vector and the optimal solution to the least time transportation problem in Step 2.
- (5) Determine the possible decrease in  $\tau$  that would result if the  $i^{\text{th}}$  ship was scheduled to one of the other ports such that its travel time would be a minimum over all the other ports. The possible decrease in  $\tau$  is the sum over all the ships in  $S$  of the possible decrease for each ship. If the possible decrease in  $\tau$  is zero, stop; otherwise

continue on to Step 6.

(6) For all the ships for which a decrease in  $T_i^j$  is possible, adjust the port requirements in accordance with the new  $T_j^i$ 's (i.e. for a particular ship  $i$ , if  $T_i^j - T_{ii}^{j*}$  is positive, where  $T_j^{i*}$  is the travel time of ship  $i$ , if it were scheduled through port  $j^*$  instead of port  $j$ , then decrease port requirements at  $j$  by one and increase port requirements at  $j^*$  by one). Return to Step 2.

This algorithm successfully determines a new set of port requirements for the Bases-to-Ports problem such that the total travel time, for iteration  $(K+1)$  is less than  $\tau$  for iteration  $(K)$ . This algorithm does, in fact, attempt to find the set of port requirements such that the closure (i.e.,  $\max_{i \in S} \{ T_k^j \}$ ) is minimal. This can be seen from the way the possible decrease in  $\tau$  is determined. The possible decrease in  $\tau$  is the sum of the reductions in the individual ship travel times if a different set of port requirements was used. So even though this algorithm does not explicitly minimize the maximal ship travel time and does not explicitly solve the Ships-to-Ports scheduling problem until after the algorithm has determined the best set of port requirements to use, it does generate schedule of the ships to the ports that would be near the optimal schedule of the total strategic transportation problem. Since closure for the strategic transportation problem is not determined until this algorithm has rendered the best set of port requirements in terms of the total travel time criterion, a checking procedure might be employed to determine if this solution is close to the overall minimal closure solution. To accomplish the

checking, sequentially perturb the minimal  $\tau$  set of port requirements and solve the corresponding Bases-to-Port problem and Ships-to-Ports problem for this new requirement set, continue until the minimal closure solution is found. With a limited amount of computational experience we have that this algorithm will indeed generate a solution to the total strategic transportation problem which is very close to the overall optimal, and in some cases the solution rendered was in fact the optimal solution.

An example, utilizing the total travel time concept as a criterion for altering the set of pore requirements is now presented.

GIVEN:

Number of bases = 2

Number of ports = 2

Number of ships = 6

Availability of supplies at the bases

3
3

Ship-to-Port travel time matrix

PORTS	
1	2

S H I P S	1	2	4
	2	5	3
	3	6	1
	4	3	2
	5	4	5
	6	3	5

Port-to-objective travel time vector

2
3

## ALGORITHM:

STEP 1: Use an even port requirement allocation (i.e.  $\begin{bmatrix} 3 \\ 3 \end{bmatrix}$  )

STEP 2: The solution of the least time transportation problem using an availability vector (3, 3) and a requirement vector (3,3). The Base-to-Port travel time matrix is not needed for the complete algorithm, but it is however required for step 2.

BP MATRIX		PORTS			
		1	2	1	2
B					
A	1	4	3	0	3
S					
E	2	2	5	3	0
		$c_{ij}$		$x_{ij}$	

The solution of the least time transportation problem can be expressed in matrix form, where the  $ij^{th}$  element is the number of ship loads of supplies arriving at port  $j$  in time period  $i$ .

		PORTS	
		1	2
T	1	0	0
I			
M	2	3	0
E	3	0	3

STEP 3: Assign ships 1, 5, 6 to port one, and ships 2, 3, 4 to port two.

$$\text{STEP 4: } T_i^j = SP_{ij} + d_i^j + PO_j$$

$$T_1^1 = 2 + 0 + 2 = 4$$

$$T_5^1 = 4 + 0 + 2 = 6$$

$$T_6^1 = 3 + 0 + 2 = 5$$

The delay is zero for ships 1, 5 and 6 since all supplies arrive at time two.

$$T_2^2 = 3 + 0 + 3 = 6$$

$$T_3^2 = 1 + 2 + 3 = 6$$

$$T_4^2 = 2 + 1 + 3 = 6$$

The delay experienced by ships 3, 4 is due to the arrival of supplies at time 3.

$$\begin{aligned} T &= \sum_{i \in S} \sum_{j \in P} T_i^j \\ &= T_1^1 + T_5^1 + T_6^1 + T_2^2 + T_3^2 + T_4^2 \\ &= 4 + 6 + 5 + 6 + 6 + 6 = 33 \end{aligned}$$

STEP 5: Any possible decrease in  $\tau$  would occur only if the ship(s) assigned to port 1 were assigned to port 2 and vice versa. Therefore, travel times for all ships are calculated if they were assigned to the other ports.

$$\begin{array}{lll} T_1^2 = 7 & T_3^1 = 8 & T_5^2 = 8 \\ T_2^1 = 7 & T_4^1 = 5 & T_6^2 = 8 \end{array}$$

Possible decrease in  $\tau$  equals 1, since the travel time of ship 4 could be reduced one unit if it were assigned to port 1.

STEP 6: The new requirement vector becomes  $\begin{bmatrix} 4 \\ 2 \end{bmatrix}$  .

Now return to Step 2.

For this example the algorithm does in fact terminate with the set of port requirements (i.e. (4, 2)) which will generate a ship schedule that has the overall minimal closure time. However, in general the algorithm will not determine the best set of port requirement, but it will find a set that renders a near-optimal ship schedule.



## CHAPTER IX

### CONCLUSIONS AND RECOMMENDATIONS

This research has been concerned with a study of a class of network problems, known as minimal closure problems. A strategic transportation problem has been used as a medium through which this study was conducted.

The objectives of this study were to characterize the structure of the strategic transportation problem within the framework of other network problems discussed in the literature and to develop a solution algorithm. We have observed that the strategic transportation problem can be decomposed into two subproblems. One of these subproblems is concerned with the transportation of a commodity from a set of sources to a set of destinations. And the second subproblem deals with the scheduling of a group of carriers to the same set of destinations, so that the commodity may be transferred to these carriers and transported to a common location (or objective area). The coupling link between these two subproblems is the units of the commodity, available at each destination for further transportation by the carriers. The first subproblem could be solved by a standard transportation algorithm and the second subproblem could be solved by a maximal flow algorithm. However, the overall objective of a strategic transportation problem is minimal closure (i.e. the arrival time of the last unit of commodity at the objective area must be minimal). Therefore, the first subproblem

transports all of the commodity to the destinations such that the last unit arrives in a minimal amount of time. The second subproblem must schedule the carriers to the destinations such that the arrival time of the last one at the objective area is also minimal.

### A Summary of Research Results

In order to insure minimal closure, the transportation subproblem must be solved by an algorithm which does not minimize the total time required to transport all of the commodity, but minimizes the longest time required to transport any units of the commodity. The general solution algorithm discussed in this thesis uses a series of standard transportation problems and a penalization scheme to solve this problem.

A network can be constructed to represent both the operation through time of each port (destination) and the travel of each ship to a port and from the port to the objective area. The ship scheduling subproblem is formulated as a maximal flow problem through this network. The maximal flow solution will not however insure minimal closure for this subproblem. Hinkle has developed a solution algorithm for the minimal closure, maximal flow problem (min-max path flow problem) through a network which has only upper arc capacities in it. Since lower arc capacities are required in the ship scheduling network to force flow (ships) through certain arcs, Hinkle's algorithm could not be directly applied to the ship scheduling subproblem. Two extensions of his algorithm have been made to assure its applicability to this subproblem. A constraint switching procedure has been added to his basic algorithm to eliminate the necessity having both an upper and lower

capacity constraint represented by a row in the linear programming basis of the ship scheduling subproblem. Using a constrained shortest path problem, Hinkle's algorithm finds a path through the network which tends to reduce the flow on the longest path. A labelling routine, which does not allow negative arc costs, is used to solve the constrained shortest path problem. In a network with lower arc capacities there is a possibility of obtaining negative arc costs, however, the difficulties arising from negative cost, directed cycles do not exist in the ship scheduling network since it is acyclic. An acceleration version of the labelling procedure, which takes advantage of the acyclic property of the network, was also developed in this thesis.

The second subproblem determines a schedule of the ships to the ports such that the arrival time of the last ship at the objective area is minimal. Since each ship must transport one unit of the commodity from the ports, to the objective area, the number of units available at each port is also the number of ships that must travel to the port. Therefore the minimal closure schedule rendered by the second subproblem is only optimal with respect to the set of units available at the ports. A better overall solution to the strategic transportation problem might exist for a different set of the port requirements (i.e. units available for transportation at each port). An iterative solution algorithm was developed based on changing the port requirements in such a way that an improved overall solution to the strategic transportation problem is found. A branch and bound algorithm is used to combine the transportation subproblem with the scheduling subproblem, in such a way that solution to the second subproblem will indicate how to change the port

requirements of the first subproblem in order to improve the overall minimal closure of the strategic transportation problem. Using two basic results of linear programming theory, namely the interpretation of the dual variables and the elements of the basis inverse matrix, the optimal solution of the scheduling subproblem indicates a new set (or sets) of port requirements for the transportation subproblem. If the total strategic transportation problem is solved for each of these sets of port requirements, a bound on closure time is determined for each set. The algorithm will tend to branch to the best set of port requirements in terms of this bound and the determination of new sets of port requirements will be repeated. A backtracking scheme can be used to determine if a better set of port requirement in terms of the bound exists along another portion of the tree generated by the branch and bound algorithm. If a backtracking scheme is added to this branch and bound algorithm, an optimal solution to the strategic transportation problem is guaranteed to be found.

If the backtracking scheme is eliminated from the above solution algorithm then the algorithm represents a one pass type algorithm, since all the possible sets of port requirements are not examined. However, a one pass branch and bound algorithm will be a good heuristic algorithm and it would generate a near-optimal solution to the strategic transportation problem. Another heuristic algorithm was developed which only uses the solution of the transportation subproblem for a given set of port requirements and the ship travel time information to generate a new set of port requirements. After calculating the total travel time for all of the ships for the new set of port requirements,

it is determined if this value of total travel time can be reduced by assigning a particular ship to another. And if the total travel time can be reduced the port requirements are changed accordingly and the transportation subproblem solved again using the new set of port requirements. However, if no reduction is possible by reassigning the ships, the best set of port requirements in terms of the total travel time of all of the ships has been found and the actual schedule of the ships and the closure time is found solving the second subproblem. Again, this algorithm will generate a near-optimal solution to the total strategic transportation problem.

### Other Applications

#### Multi-Commodity Flow Problems

The general multi-commodity flow problem would require both a lower and upper capacity on certain arcs of the network to force the flow of the commodities through these particular arcs. A positive lower capacity might be required on an arc leading out of particular commodity's source node to insure that a specified amount of the commodity move through the network to its sink node. The multi-commodity flow problem could be stated as maximize the flow of all the commodities through the network such that the arc capacities are not violated. The mathematical formulation of this problem in arc path form is exactly the maximal flow problem through a capacitated network discussed in Chapter III. Since each path through the network of the multi-commodity problem represents a route over which the flow of that commodity can be moved, and if all the path through the network were enumerated for all

the commodities the sum of the flow on the paths using a particular arc must not violate its capacities, either upper or lower. If the multi-commodity problem was to be solved by linear programming techniques the basis would require a row for each lower capacity constraint and for each upper capacity constraint. Since both constraints for a particular arc are not needed to bound the flow at the same time, the constraint switching procedure developed in Chapter V could be applied to this problem to allow the reduced basis to accurately represent all of the constraints in the multi-commodity flow problem.

#### Other Transportation Problems

Transportation problems in other areas besides the military can be modeled as a strategic transportation problem. One area might be the perishable food industry, where moving the crops to market or to a processing plant must be accomplished in such a way that all of the product would reach its final destination before spoilage destroyed it. The transportation of the crops from the farms to a set of collection centers could be modeled by the least-time transportation problem. If the food processor had a group of trucks for instance, that could be scheduled to any of the collection centers, then the scheduling subproblem could be used to determine a minimal closure schedule of these trucks. In addition, one of the solution algorithms developed in Chapter VIII of this thesis could be implemented, so that the overall minimal closure solution could be found. Now the food processor would be in the position to determine the minimal amount of time required to transport all the crops from the farms to his plant and this information could then be used to accurately schedule the processing operation

at the plant.

The general structure of the strategic transportation problem could be used in modeling other transportation problems, such as a transshipment type problem. A transshipment problem could consist of set of locations which act as destinations with respect to one stage of the problem, and act as sources in a second stage. In each stage there could exist a difference mode of transportation, one of which has an unlimited availability and low operating costs and the other might have a limited availability and high operating costs. The total problem could be decomposed into subproblems with the coupling link being the requirements for the commodity at the common set of locations. This decomposition would allow one of the subproblems to be modeled with the transportation of the commodity as the primary objective and the scheduling of the carriers to be implicitly accomplished through the solution algorithm. The other subproblem could be modeled with scheduling of the carriers as the primary objective. Therefore a multi-stage transportation could be modeled as separate subproblem with different objectives and yet accomplishing the overall objective of the problem, that is to transport the commodity from the original sources to the final destinations.

#### Extensions and Areas of Further Research

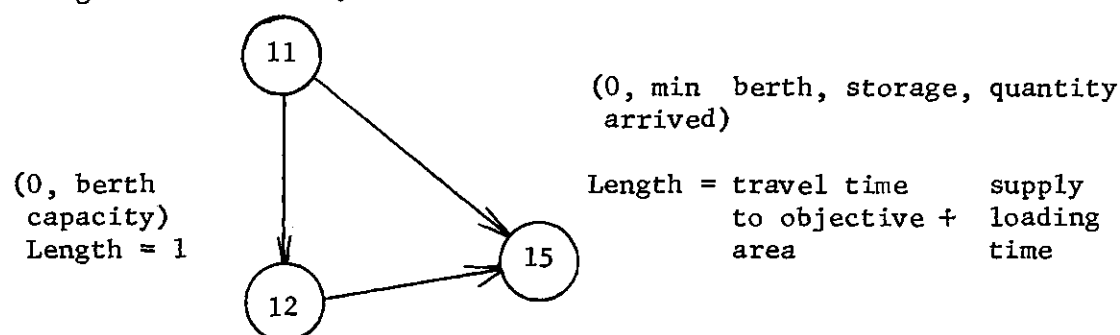
The strategic transportation problem discussed in this thesis has been formulated in view of what the military anticipates its supply systems will be like in the future. However there exists a need for an accurate model and solution algorithm for the military supply systems of today. There are a number of characteristic of the future supply

system which are not present in today's systems. These characteristics might be viewed as simplifying assumptions in the current supply systems. It would seem, however, that the structure and the algorithm could be extended and/or modified to model a system in which these special characteristics did not exist.

Let us examine a number of these special characteristics and indicate how the strategic transportation model and/or the solution algorithm might change in the absence of them. The network constructed in Chapter III to model the ship scheduling subproblem did not include either supply storage or berth capacities, since the port's operation only involved the transfer of standardized units of supplies from one mode of transportation to another (i.e. to the ships). A berth capacity would limit the number of ships that could be loaded with supplies a time unit in the port's operation. A storage capacity at a port would also limit the number of ships that could be loaded, and it also limits the number of ships that could leave a port during one time unit. The arc between two subnodes of a port's temporal expansion represents the passage of one time unit. A berth capacity can be used as an upper arc capacity to limit the number of ships that stay in the port waiting to be loaded. The arc between a subnode and a port's sink node represents a loaded ship leaving the port and traveling to the objective area. An upper capacity on this arc is maximum number of loaded ships that can leave a port during one time unit. To accurately model a port with storage and berth capacities this upper capacity must be the minimum of these two capacities. However, if the quantity of supplies that have arrived at the port up until this time unit is less than the



minimum of the storage and berth capacities, then only that number of loaded ships could possibly leave the port. Therefore the upper capacity of these arcs must be the minimum of all three values. There would also exist a time associated with the actual loading of the ships. However, this time would be the same for each ship loaded at a port, so it could simply be added to the lengths of the arcs connecting the subnodes and the port sink nodes. Graphically the arcs of the port portion of the ship scheduling network would now have the capacities and lengths shown below.



Only one difficulty seems apparent if both berth and storage capacities are added to the network representation of a port's operation; and it concerns the representation of ships waiting to enter a port for loading when all of the berths are full. This situation arises if the berth capacity is smaller than the number of ships that could be scheduled to arrive at a port during a time unit and no supplies will arrive at the port until some future time period. No simple way of reconstructing the network to handle this situation of out-of-port waiting has been found. So further study would be required if the basic network, constructed in Chapter III is to be modified for berth and storage capacities.

Another characteristic of the strategic transportation model developed in this thesis is that all of the ships are the same size (i.e. they have the same load carrying capacity) and travel at the same speed. Our model changes drastically if these ship characteristics are not present. In the context of the Bases-to-Ports transportation problem, the supplies would no longer be transported in standardized ship loads, but they would be moved as units of the individual types of supplies. This transportation problem becomes a multi-commodity transportation problem with each commodity being a different type of supply. Since there are no capacities on the travel links in the Bases-to-Ports problem, the minimal cost solution (i.e.  $\min \sum c_{ij} x_{ij}$ ) to the multi-commodity transportation problem can be determined by solving a series of single commodity, standard transportation problems. If the penalization scheme discussed in Chapter II is used in the solution of each single commodity transportation problem, then we would be solving a series of min-max (least time) transportation problems. And thus a min-max solution for this multicommodity transportation problem could be found. If the individual types of supplies are transported from the bases to the ports, instead of ship load units of supplies, there are no real difficulties in determining the least time solution for the Bases-to-Ports problem, except the computational effort required has increased significantly.

The Ships-to-Ports subproblem of the strategic transportation problem was easily formulated as a min-max path flow problem. The network constructed in Chapter II, represented each ship's travel to a port to take on supplies and its travel to the objective area. Since

each ship could only carry one ship load of supplies, the flow through the network could also be interpreted as ship loads of supplies moving to the objective area. Arc capacities were added to the network to both force and limit flow through the network. The arcs connecting each port sink node and the super-sink were added as driving arcs, in the sense that their upper and lower capacities equaled the number of ships that must move through each port, which is also the number of ship loads of supplies that were available at each port for further transportation. Both the flow and the arc capacities were in common units and the network represented a single commodity problem, which the min-max path flow algorithm, with the modifications described in this work, could easily be solved. Now however, there are different sizes of ships and different types of supplies arriving at the ports. Each ship no longer carries a standard load of supplies, but it could carry any combination of supplies as long as the carrying capacity was not exceeded. There no longer exists a common unit in either the flow or the arc capacities. And now the Ships-to-Ports scheduling problem is also a multi-commodity problem. Since neither the network representation nor the solution algorithm developed in this thesis, can be applied to this problem further study, directed at how this multi-commodity situation might be represented on a network and how the solution algorithm might be modified, is required.

In Chapter VIII, an iterative method, using a branch and bound algorithm, was developed for the solution of the total strategic transportation problem. It was indicated that the solution to the ship scheduling problem could be used to generate a new set of port

requirements, such that the closure time for the total problem is lower than the closure time for the best previous set of port requirements. Implicit in the determination of the closure time for the new set of port requirements is the complete solution of both subproblems using this new set. By changing the port requirements, the min-max solution to the Bases-to-Ports transportation problem will indicate a new set of arrival times of the supplies at the ports. Since the quantities and the arrival times of the supplies at each port are used as arc capacities in the ship scheduling network, the new solution of the Bases-to-Ports problem will dictate certain changes in the arc capacities. These arc capacities are used as the "b" values in the linear programming formulation of the ship scheduling subproblem. Instead of completely resolving this subproblem with the new "b" values, a sensitivity analysis of the previous optimal solution to the ship scheduling problem can be performed in light of these new "b" values. Since the optimal solution in terms of the variables (paths) that are in the basis, will not change if the "b" values are changed, however, the value of these variables might. The new variable values can be determined by simply updating the new b vector by matrix multiplication with the current basis inverse (i.e.  $b = B^{-1}b$ ). If all of the updated "b" values remain non-negative then we have obtained the new solution to the ship scheduling problem without completely resolving it. However, if any of the new "b" values become negative during the updating process, then this new solution is infeasible. Therefore any variable with a negative value must be removed from the basis. One method to easily accomplish this removal would be the dual simplex algorithm. This same type of sensitivity analysis could be used in determining the new

solution to Bases-to-Ports subproblem if it was solved as a linear programming problem. However, it is solved using the transportation algorithm. Some study would be required to determine if there did exist a method to use the previous solution to this subproblem in connection with changes in port requirements, to determine the least-time solution without completely resolving the problem. In general the use of sensitivity analyses could greatly reduce the computational effort required in the complete solution algorithm for the strategic transportation problem.

## APPENDICES

## APPENDIX A

## PROGRAM LISTING

CODE FOR STRATEGIC TRANSPORTATION PROBLEM

DATE: 101473      TIME: 124218

```

      DIMENSION BP(20,20),A(20),B(20),TB(20),TAB(20,20)
      DIMENSION ITAB(20,20),INDEX(20,2),NROW(20),NCOL(20),SUPPLY(20,20)
      DIMENSION SP(20,20),PO(20)
      INTEGER BP,BASES,PORTS,SHIPS,ROWS,COLUMN,TEST,SUPPLY,SP,PO
      EQUIVALENCE (TAB,ITAB)
      COMMON /HELP/ SUPPLY,BP,SP,PO,1B
      COMMON /HELP/ BASES,PORTS,SHIPS,MAXTIM
C**** REQUIRED INPUT:
C          NUMBER OF BASES
C          NUMBER OF PORTS
C          NUMBER OF SHIPS
C          AVAILABILITIES AT PORTS
C          REQUIREMENTS AT PORTS
C          BASE TO PORT TIME MATRIX
C          SHIP TO PORT TIME MATRIX
C          PORT TO OBJECTIVE TIME VECTOR
      READ(5,100) BASES,PORTS,SHIPS
      READ(5,100) (A(I),I=1,BASES)
      READ(5,100) (B(J),J=1,PORTS)
      READ(5,100) ((BP(I,J),J=1,PORTS),I=1,BASES)
      READ(5,100) ((SP(I,J),J=1,PORTS),I=1,SHIPS)
      READ(5,100) (PO(J),J=1,PORTS)
100      FORMAT( )
C          WRITE(6,110) BASES,PORTS,SHIPS
C110      FORMAT(315)
X          WRITE(6,111) (A(I),I=1,BASES)
X111      FORMAT(2F6.2)
C          WRITE(6,112) (B(J),J=1,PORTS)
C112      FORMAT(2F6.2)
          WRITE(6,113) ((BP(I,J),J=1,PORTS)
C113      FORMAT(215)
C          WRITE(6,113) ((SP(I,J),J=1,PORTS), I=1,SHIPS)
C          WRITE(6,113) (PO(J),J=1,PORTS)
          WRITE(6,114)
114      FORMAT(' INTERMEDIATE TABLEAUS FOR LTT')
          DO 10 J=1,PORTS
10          IB(J)=B(J)
C**** NORTHWEST CORNER RULE

```

```

      I=1
      J=1
      ZERO=.00
      EPS=.01
      INF=99999
      MAXTIM=-INF
      K=1
120  IF(A(I).LE.B(J)) GO TO 130
      TAB(I,J)=B(J)
      A(I)=A(I)-B(J)
      IF(BP(I,J).LT.MAXTIM) GO TO 125
      MAXTIM=BP(I,J)
      NROW(K)=I
      NCOL(K)=J
125  IF(A(I).GE.ZERO+EPS) GO TO 135
      I=I+1
      TAB(I,J)=EPS
135  J=J+1
140  IF(I.GT.BASES.OR.J.GT.PORTS) GO TO 160
      GO TO 120
130  TAB(I,J)=A(I)
      B(J)=B(J)-A(I)
      IF(BP(I,J).LT.MAXTIM) GO TO 145
      MAXTIM=BP(I,J)
      NROW(K)=I
      NCOL(K)=J
145  IF(B(J).GE.ZERO+EPS) GO TO 150
      J=J+1
      TAB(I,J)=EPS
150  I=I+1
      GO TO 140
C****      NORTHWEST CORNER SOLUTION
160  WRITE(6,109)
109  FORMAT('O TABLEAU FROM NORTHWEST CORNER RULE')
      WRITE(6,105)((TAB(I,J),J=1,PORTS),I=1,BASES)
105  FORMAT(2F6.2)
C****      LEAST TIME ALGORITHM
161  LL=K
      DO 600 L=1,LL
      MMM=PORTS+BASES
165  DO 170 I=1,MMM
      DO 170 J=1,2
170  INDEX(I,J)=0
      INDEX(1,1)=NROW(L)
      INDEX(1,2)=NCOL(L)
      MROW=NROW(L)
      MCOL=NCOL(L)
      K=1
C****      SEARCH FOR ROW CELLS
      DO 180 J=L,PORTS
      IF(TAB(MROW,J).LT.EPS) GO TO 180

```



```

        IF(MCOL.EQ.J) GO TO 180
        K=K+1
        INDEX(K,1)=MROW
        INDEX(K,2)=J
180    CONTINUE
        ROWS=K
C***** SEARCH FOR COLUMN CELLS
        DO 190 I=1, BASES
        IF(TAB(I,MCOL).LT.EPS) GO TO 190
        IF(I.EQ.MROW) GO TO 190
        K=K+1
        INDEX(K,1)=I
        INDEX(K,2)=MCOL
190    CONTINUE
        COLUMN=K
C      WRITE(6,500)((INDEX(I,J),J=1,2),I=1,K)
C500    FORMAT(215)
C***** SEARCH FOR PATHS TO CHANGE ALLOCATION TO MAXTIM CELL
        IF(ROWS.EQ.1) GO TO 215
        DO 200 K=2,ROWS
        M=INDEX(K,2)
        DO 210, I=1,BASES
        IF(I.EQ.MROW) GO TO 210
        IF(TAB(I,M).LT.ZERO+EPS) GO TO 210
        IF(BP(I,MCOL).GE.MAXTIM) GO TO 210
        TEST=0
        GO TO 240
210    CONTINUE
200    CONTINUE
C***** NO PATH FOUND YET!
        IF(COLUMN.EQ.ROWS) GO TO 600
215    ROWS=ROWS+1
        DO 220 K=ROWS,COLUMN
        M=INDEX(K,1)
        DO 230 J=1,PORTS
        IF(J.EQ.MCOL) GO TO 230
        IF(TAB(M,J).LT.ZERO+EPS) GO TO 230
        IF(BP(MROW,J).GE.MAXTIM) GO TO 230
        TEST=1
        GO TO 240
230    CONTINUE
220    CONTINUE
C***** NO PATH FOUND!
        GO TO 600
240    IF(TEST.EQ.0)GO TO 300
C***** COLUMN CHANGE
        IF(TAB(MROW,MCOL).LE.TAB(M,J)) GO TO 250
        CHANGE=TAB(M,J)
        GO TO 255
250    CHANGE=TAB(MROW,MCOL)
255    TAB(MROW,MCOL)=TAB(MROW,MCOL)-CHANGE

```

```

        TAB(M,J)=TAB(M,J)-CHANGE
        TAB(M,MCOL)=TAB(M,MCOL)+CHANGE
        TAB(MROW,J)=TAB(MROW,J)+CHANGE
        IF(TAB(MROW,MCOL).LT.ZERO+EPS) GO TO 600
        WRITE(6,502)
        WRITE(6,501)((TAB(I,J),J=1,PORTS),I=1,BASES)
501    FORMAT(2F6.2)
        GO TO 165
C****  ROW CHANGE
300    IF(TAB(MROW,MCOL).LE.TAB(I,M)) GO TO 260
        CHANGE=TAB(I,M)
        GO TO 265
260    CHANGE=TAB(MROW,MCOL)
265    TAB(MROW,MCOL)=TAB(MROW,MCOL)-CHANGE
        TAB(I,M)=TAB(I,M)-CHANGE
        TAB(MROW,M)=TAB(MROW,M)+CHANGE
        TAB(IMCOL)=TAB(I,MCOL)+CHANGE
        IF(TAB(MROW,MCOL).LT.ZERO+EPS) GO TO 600
        WRITE(6,502)
        WRITE(6,501)((TAB(I,J),J=1,PORTS),I=1,BASES)
        GO TO 165
600    CONTINUE
        MAX=BP(MROW,MCOL)
C****  RECALCULATE MAXTIM
305    MAXTIM=-INF
        WRITE(6,502)
502    FORMAT('O OTHER TABLEAUS')
        WRITE(6,501)((TAB(I,J),J=1,PORTS),I=1,BASES)
        DO 310 I=1,BASES
        DO 320 J=1,PORTS
        IF(TAB(I,J).LT.ZERO+EPS) GO TO 320
        IF(BP(I,J).LT.MAXTIM) GO TO 320
        MAXTIM=BP(I,J)
320    CONTINUE
310    CONTINUE
        IF(MAXTIM.EQ.MAX) GO TO 400
        K=0
        DO 325 I=1,BASES
        DO 330 J=1,PORTS
        IF(TAB(I,J).LT.ZERO+EPS) GO TO 330
        IF(BP(I,J).NE.MAXTIM) GO TO 330
        K=K+1
        NROW(K)=I
        NCOL(K)=J
330    CONTINUE
325    CONTINUE
        GO TO 161
C****  CONVERSION OF FLOATING PT. INFO. TO INTEGER INFO.
400    DO 405 I=1,BASES
        DO 405 J=1,PORTS

```

```

      TAB(I,J)=TAB(I,J)+(4.*EPS)
      ITAB(I,J)=TAB(I,J)
405  CONTINUE
      DO 410 I=1,MAXTIM
      DO 410 J=1,PORTS
410  SUPPLY(I,J)=0
      DO 415 I=1,BASES
      DO 420 J=1,PORTS
      IF(ITAB(I,J).LE.0) GO TO 420
      M=BP(I,J)
      SUPPLY(M,J)=ITAB(I,J)
420  CONTINUE
415  CONTINUE
C**** OPTIMAL SOLUTION
      WRITE(6,106)
106  FORMAT('OPTIMAL SOLUTION TO LIT',/,'FINAL TABLEAU')
      WRITE(6,107)((ITAB(1,J),J=1,PORTS),I=1,BASES)
      WRITE(6,108)
108  FORMAT('SUPPLY MATRIX')
      WRITE(6,107)((SUPPLY(I,J),J=1,PORTS),I=1,MAXTIM)
107  FORMAT(215)
      CALL CONSTR
      END

      SUBROUTINE CONSTR
      DIMENSION INPUT(100,5),BP(20,20),SP(20,20),PO(20),SUPPLY(20,20)
      DIMENSION SUM(20),REQUIR(20),NODEST(25,25)
      DIMENSION B1(100),E(100),UC(100),LC(100),LENGTH(100),Y(100)
      INTEGER B1,E,UC,Y,SOURCE
      INTEGER BP,SP,PO,SUPPLY,SUM,REQUIR,BASES,PORTS,SHIPS,TIME
      INTEGER ARCS,TEMPOR,COUNTR,X
      COMMON /HELP/ SUPPLY,BP,SP,PO,REQUIR
      COMMON /HELP2/ BASES,PORTS,SHIPS,TIME
      COMMON /NETWOR/ NODES,ARCS,SOURCE,B1,E,LENGTH,UC,LC,Y
      WRITE(6,101)BASES,PORTS,SHIPS
101  FORMAT('NUMBER OF BASES: ',15,/,' NUMBER OF PORTS: ', 15,
1/, ' NUMBER OF SHIPS: ',15)
      INF=99999
C**** ARCS FROM PORTS SINKS TO SUPER SINK
      NODES=PORTS+1
      DO 110 I=2,NODES
      ARCS=I-1
      INPUT(ARCS,1)=1
      INPUT(ARCS,2)=1
      INPUT(ARCS,3)=0
      INPUT(ARCS,4)=REQUIR(ARCS)+5
      INPUT(ARCS,5)=REQUIR(ARCS)
110  CONTINUE
C**** DETERMINE SIZE OF TEMPORAL EXPANSION
      DO 120 J=1,PORTS
      TEMPOR=-INF

```

```

        IF(SUPPLY(I,J).EQ.0) GO TO 130
        IF(TEMPOR.GE.I) GO TO 130
        TEMPOR=I
130    CONTINUE
        DO 140 I=1,SHIPS
        IF(SP(I,J).LE.TEMPOR) GO TO 140
        TEMPOR=SP(I,J)
140    CONTINUE
C***** CALCULATE CUMMULATIVE SUPPLY ARRIVALS FOR EACH TIME PERIOD
        DO 150 I=1,TEMPOR
150    SUM(I)=0
        IHELP=TEMPOR+1
        DO 160 L=2,IHELP
        IF(L-1.GT.TIME) GO TO 165
        LL=L-1
        SUM(L)=SUM(LL)+SUPPLY(LL,J)
        GO TO 160
165    LL=L-1
        SUM(L)=SUM(LL)
160    CONTINUE
C***** ARCS FROM TEMPORAL EXPANSION TO PORT SINKS
        KK=NODES+1
        KKK=NODES+TEMPOR
        I=2
        DO 170 K=KK,KKK
        ARCS=ARCS+1
        INPUT(ARCS,1)=K
        INPUT(ARCS,2)=J+1
        INPUT(ARCS,3)=PO(J)
        INPUT(ARCS,4)=SUM(I)
        INPUT(ARCS,5)=0
        L=I-1
        NODEST(L,J)=K
        I=I+1
170    CONTINUE
C***** CONNECTING ARCS IN TEMPORAL EXPANSION
        NODES=NODES+TEMPOR
        DO 180 K=KK,NODES
        ARCS=ARCS+1
        IF(K.EQ.NODES) GO TO 180
        INPUT(ARCS,1)=K
        INPUT(ARCS,2)=K+1
        INPUT(ARCS,3)=1
        INPUT(ARCS,4)=SUM(TEMPOR+1)
        INPUT(ARCS,5)=0
180    CONTINUE
C***** ADD EXTRA NODES AND ARCS TO AVOID DUPLICATE SHIP ARRIVALS
        ARCS=ARCS-1
        NNN=ARCS-TEMPOR+1
        NN=ARCS-2*(TEMPOR=1)

```

```

        COUNTR=0
        DO 200 N=NN,NNN
        IF(INPUT(N,4).EQ.0) GO TO 200
        IF(N.EQ.NNN) GO TO 191
        L=N+1
        IF(INPUT(N,4).EQ.INPUT(L,4) GO TO 190
191    IF(COUNTR.EQ.0) GO TO 200
        II=N-COUNTR
        DO 195 I=II,N
        INPUT(I,2)=NODES+1
195    CONTINUE
        ARCS=ARCS+1
        INPUT(ARCS,1)=NODES+1
        INPUT(ARCS,2)=J+1
        INPUT(ARCS,3)=0
        INPUT(ARCS,4)=INPUT(N,4)
        INPUT(ARCS,5)=0
        COUNTR=0
        NODES=NODES+1
        GO TO 200
190    COUNTR=COUNTR+1
200    CONTINUE
120    CONTINUE
C**** ARCS FROM SHIPS TO PORTS
        DO 210 I=1,SHIPS
        DO 220 J=1,PORTS
        ARCS=ARCS+1
        INPUT(ARCS,1)=NODES+1
        X=SP(I,J)
        INPUT(ARCS,2)=NODEST(X,J)
        INPUT(ARCS,3)=X
        INPUT(ARCS,4)=1
        INPUT(ARCS,5)=0
220    CONTINUE
210    CONTINUE
C**** ARCS FOR SUPER SOURCE
        NN=NODES+1
        NNN=NODES+SHIPS
        NODES=NODES+SHIPS+1
        DO 230 N=NN,NNN
        ARCS=ARCS+1
        INPUT(ARCS,1)=NODES
        INPUT(ARCS,2)=N
        INPUT(ARCS,3)=0
        INPUT(ARCS,4)=1
        INPUT(ARCS,5)=0
230    CONTINUE
        SOURCE=NODES
        WRITE(6,102) NODES,ARCS,SOURCE
102    FORMAT(' NUMBER OF NODES: ',15,/, ' NUMBER OF ARCS: ',15,

```

```

1/,' SOURCE IS: ',15)
C      WRITE(6,103)
C103   FORMAT(' BEGINNING ENDING LENGTH UPPER LOWER')
C      WRITE(6,104)((INPUT(I,J),J=1,5),I=1,ARCS)
C104   FORMAT(16,317,16)
      DO 700 K=1,ARCS
      B1(K)=INPUT(K,1)
      E(K)=INPUT(K,2)
      LENGTH(K)=INPUT(K,3)
      UC(K)=INPUT(K,4)
      LC(K)=INPUT(K,5)
700    CONTINUE
      DO 701 J=1,ARCS
701    Y(J)=0
      CALL START
      RETURN
      END

      SUBROUTINE START
      DIMENSION B1(100),E(100),UC(100),XB(100),BASIS(100,100),LC(100)
      DIMENSION LENGTH(100),BINV(100,100),LBASIS(100),ACT(100),X(100)
      DIMENSION PATH (100)
      COMMON /NETWOR/NODES,ARCS,SOURCE, B1,E,LENGTH,UC,LC,X
      COMMON /SOLN/ DIM,BINV,XB,BASIS,LBASIS,ACT
      COMMON D,OUT
      INTEGER PATH
      INTEGER B1,E,ARCS,SOURCE,DIM,ACT,X,UC
C
C**** LEXICON OF IMPORTANT VARIABLES
C NODES      NUMBER OF NODES IN NETWORK
C ARCS       NUMBER OF ARCS IN NETWORK
C BASIS      BASIS MATRIX
C BINV       INVERSE OF BASIS MATRIX
C DIM        SIZE OF MATRICES
C B1         LIST OF BEGINNING NODES FOR THE ARCS
C E          LIST OF ENDING NODES FOR THE ARCS
C LENGTH     LENGTH OF THE ARCS
C LC         LOWER CAPACITY OF FLOW ON THE ARCS
C UC         UPPER CAPACITY OF FLOW ON THE ARCS
C X          FLOW ON THE ARCS
C XB         FLOW ON THE PATHS
C LBASIS     LENGTH OF PATHS IN THE BASIS
C ACT        ACTIVE CONSTRAINTS, EITHER UPPER OR LOWER
C PATH       VECTOR CONTAINING ARCS IN THE PATH
C STORE      VECTOR USED IN CALCULATING UPDATE,STORES PATH VECTOR
C UPDATE     UPDATED ENTERING VECTOR
C PRICE      CALCULATED ARC COSTS PASSED TO CONSTRAINED SHORTESTRPATH
C SIZE       NUMBER OF ARCS USED TO MAKE UP PATH
C LSTAR      LENGTH OF LONGEST PATH IN BASIS WITH FLOW ON IT
C

```

```

C**** NODE 1 IS THE SINK
C**** PATHS ARE ASSUMED TO HAVE POSITIVE LENGTHS
C**** AND SLACKS HAVE ZERO LENGTH
C**** LAST NODE IS THE SOURCE
C**** ANY ARCS WITH UPPER AND LOWER
C**** BOUNDS OF ZERO MUST BE REMOVED FROM THE NETWORK
C**** PROGRAM WILL NOT HANDLE ARCS WITH EQUAL LOWER
C**** AND UPPER BOUNDS THUS ONE MUST BE CHANGED !
C
      INF=999999
C      READ(5,1111)(CA(I),I=1,30)
C1111  FORMAT(30A1)
C      WRITE(6,1112)(CA(I),I=1,30)
C1112  FORMAT(1X,30A1)
      WRITE(6,1000)
      1000 FORMAT(' NODES,ARCS, SOURCE?')
      WRITE(6,1001)
      1001 FORMAT(' ARC DATA?')
      DIM=ARCS+1
C**** PREPROCESSOR TO REMOVE ARCS WITH UC=0
      K=0
      DO 100 J=1,ARCS
      IF(UC(J).LE.0) GO TO 100
      K=K+1
      B1(K)=B1(J)
      E(K)=E(J)
      LENGTH(K)=LENGTH(J)
      UC(K)=UC(J)
      LC(K)=LC(J)
      100  CONTINUE
      ARCS=K
      DIM=ARCS+1
C**** INITIALIZE RETURN ARC DATA
      B1(DIM)=1
      E(DIM)=NODES
      LENGTH(DIM)=-999999
      UC(DIM)=999999
      LC(DIM)=0
      X(DIM)=0
      CALL FIRST
      CALL MATINV(DIM)
      IF(OUT.LT.1.)GO TO 310
      WRITE(6,105)
      105  FORMAT(' ERROR HAS OCCURRED- BASIS WRONG')
      GO TO 999
      310  CONTINUE
      WRITE(6,6000) NODES
      6000  FORMAT('NUMBER OF NODES IN THE NETWORK IS', 15)
      WRITE(6,6001)ARCS
      6001  FORMAT(' NUMBER OF ARCS IN THE NETWORK IS', 16)
      WRITE(6,6002)

```

```

6002  FORMAT(' THE FOLLOWING ARC DATA IS USED: ',
1/, ' BEGINNING NODE ENDING NODE LENGTH UPPER LOWER CAPACITY')
      DO 6005 J=1, ARCS
        WRITE(6,6004) B1(J),E(J),LENGTH(J),UC(J),LC(J)
6004  FORMAT(5I10)
6005  CONTINUE
      WRITE(6,307) XB(1)
307   FORMAT('MAX FLOW =',F7.2)
      WRITE(6,304)
304   FORMAT('ARC  ACTIVE  PATH  RIGHT HAND',/,
            TYPE      LENGTH SIDE VECTOR')
      WRITE(6,306)(J,ACT(J),LBASIS(J+1),XB(J+1),J=1,ARCS)
306   FORMAT(1X,13,4X,6X,13,7X,F7,2)
      WRITE(6,108) D
108   FORMAT(' THE DETERMINANT OF BASIS IS',F6.2)
666   CALL REVISE(LENGTH,UC,LC)
      CALL PRINT
      WRITE(6,3001)
3001  FORMAT('          OPTIMAL SOLUTION',/,
*' BASIS FLOW LENGTH PATH')
      DO 3050 I=2,DIM
        IF(LBASIS(I).EQ.0) GO TO 3060
        L=0
        K=NODES
3070  L=L+1
        DO 3080 J=2,DIM
          IF(ABS(BASIS(J,I)).LE..5)GO TO 3080
          KK=B1(J-1)
          IF(KK,EQ,K) GO TO 3090
3080  CONTINUE
C**** ERROR
3090  PATH(L)=J-1
        K=E(J-1)
        IF(K.NE.1)GO TO 3070
        WRITE(6,4000)I,XB(1),LBASIS(I),(PATH(J),J=1,L)
4000  FORMAT(16,F6.2,I8,2X,10I4,5(/,22X,10I4))
        GOTO 3050
3060  II=I-1
        WRITE(6,4001) I,XB(I),LBASIS(I),II
4001  FORMAT(16,F6.2,I8,2X,14)
3050  CONTINUE
999   RETURN
      END

```

# SUBROUTINE FIRST

```

DIMENSION B(100),LS(101),XL(100),XU(100),X(100)
DIMENSION BASIS(100,100),LBASIS(100),ACT(100),PATH(50)
DIMENSION BINV(100,100),XB(100)
DIMENSION P(100),Q(100),C(100)
COMMON F1,F2,PI(100),D(100),U(100),R(100),NC,LV
COMMON/NETWOR/M,ARCS,SOURCE,P,Q,C,XU,XL,B

```



```

COMMON/SOLN/N,BINV,XB,BASIS,LBASIS,ACT
INTEGER F1,F2,P,Q,CB,C,PI,X,XU,XL,D,U,R,B,E,T,S
INTEGER SP,SM,AP,AM,W,RR,ACT,ARCS,SOURCE
DO 430 I=1,N
  U(I)=0
430 R(I)=0
  RR=1000000000
C  INITIALIZATION
  DO 12 J=1,N
    K=P(J)
    I=Q(J)
    X(J)=XL(J)
    B(K)=B(K)-XL(J)
    B(I)=B(I)+XL(J)
12  CONTINUE
    W=N
    SM=0
    SP=0
    NI=N+SP
    NZ=NT+1
    NTT=NT
    AP=0
    AM=0
    DO 19 I=1,M
      IF(B(I).LT.0) GO TO 19
      W= W+1
      AP=AP+1
      P(W)=I
      XL(W)=0
      XU(W)=0
      X(W)=B(I)
      PI(I)=0
      C(W)=0
      D(I)=W
19  CONTINUE
      DO 89 I=1,M
        IF(B(I).GE.0) GO TO 89
        W=W+1
        AM=AM+1
        P(W)=I
        XL(W)=0
        XU(W)=RR
        X(W)=-B(I)
        PI(I)=-RR
        C(W)=RR
        D(I)=W
89  CONTINUE
        NCZ=1
C  INITIALIZATION COMPLETE
17  NTTT=NTTT+AP

```

```

      NTTTT=NTTT+AM
      ITER=0
200  IF(J.GE.N)J=0
      IFP=0
      ITER=ITER+1
      F1=1
      DO 21 JC=1,N
        J=J+1
        K=P(J)
201  L=Q(J)
        CB=C(J)-P1(K)+PI(L)
        IF(CB.GT.O.AND.X(J).EQ.XU(J).OR.CB.LT.O.AND.X(J).EQ.XL(J)) GO TO
1202
        GO TO 211
202  IF(IFP.NE.O) GO TO 203
        IFP=1
        ICMIN=CB
        JST=J
        GO TO 221
203  IF(IABS(ICMIN).GE.IABS(CB)) GO TO 221
        IFP=IFP+1
        ICMIN=CB
        JST=J
221  IF(J.EQ.N)J=0
        IF(IFP.EQ.NCZ) GO TO 205
21  CONTINUE
        IF(IFP.GT.O) GO TO 205
        J=N
        IF(SP.EQ.O) GO TO 432
        DO 22 JC=1,SP
          J=J+1
          K=P(J)
          CB=C(J)-PI(K)
          IF(CB.GT.O.AND.X(J).EQ.XU(J).OR.CB.LT.O.AND.X(J).EQ.XL(J))GO TO 2
0
22  CONTINUE
432  IF(SM.EQ.O) GO TO 431
        DO 23 JC=1,SM
          J=J+1
          K=P(J)
          CB=C(J)+PI(K)
          IF(CB.GT.O.AND.X(J).EQ.XU(J).OR.CB.LT.O.AND.X(J).EQ.XL(J))GO TO 2
0
23  CONTINUE
431  GO TO 100
205  J=JST
        CB=ICMIN
20  IF(CB.GT.O) GO TO 25
        B(1)=1
        GO TO 86

```

```

25  B(1)=-1
86  I=1
    K=P(J)
    LS(1)=J
    IF(J.LE.NT.OR.(J.GT.NTT.AND.J.LE.NTTT))GO TO 26
    GO TO 76
26  I=2
30  LS(I)=D(K)
    IF(LS(I).GT.N) GO TO 27
    IB=LS(I)
    IF(P(IB).EQ.K) GO TO 28
    B(I)=-B(1)
    K=Q(IB)
    GO TO 29
28  B(I)=B(1)
    K=Q(IB)
29  I=I+1
    GO TO 30
27  IF(LS(I).LE.NT.OR.(LS(I).GT.NTT.AND.LS(I).LE.NTTT))GO TO 31
    B(I)=-B(1)
    GO TO 32
31  B(I)=B(1)
32  IF(J.GT.N) GO TO 33
    K=Q(J)
76  IL=I
36  I=I+1
    LS(I)+D(K)
    IF(LS(I).GT.N) GO TO 34
    IB=LS(I)
    IF(Q(IB).EQ.K) GO TO 6
    B(I)=-B(1)
    K=Q(IB)
    GO TO 36
6   B(I)=B(1)
    K=P(IB)
    GO TO 36
34  IF(LS(I).EQ.LS(IL)) GO TO 60
    IF(LS(I).LE.NT.OR.(LS(I).GT.NTT.AND.LS(I).LE.NTTT))GO TO 37
    B(I)=8(1)
    GO TO 33
37  B(I)=-B(1)
C   TWO ROOTED TREES
33  E=XU(J)-XL(J)
    NI=1
    K=1
38  K=K+1
    IB=LS(K)
    IF(B(K).GT.0) GO TO 39
    T=XU(IB)=X(IB)

```

```

39 T=X(IB)-XL(IB)
40 IF(T.GE.E) GO TO 41
    E=T
    LV=LS(K)
    NI=K
    IB=P(LV)
    IF(D(IB).EQ.LV) GO TO 42
    NC=P(LV)
    GO TO 43
42 NC=Q(LV)
43 F2=1
41 IF(K.LT.1) GO TO 38
    IF(LV.LE.N) GO TO 44
    NC=P(LV)
    F2=2
44 IF(E.EQ.0) GO TO 45
    B(1)=-B(1)
    DO 46 K=1,1
    IB=LS(K)
46 X(IB)+X(IB)-B(K)*E
45 IF(NI.EQ.1) GO TO 200
    IF(NI.GT.IL) GO TO 48
    CALL LBC(J,P(J))
    K=P(J)
    L=Q(J)
    PI(K)=C(J)+P(L)
    CALL PIC(K)
    GO TO 200
48 CALL LBC(J,Q,(J))
    K=P(J)
    L=Q(J)
    PI(L)=PI(K)-C(J)
    CALL PIC(L)
    GO TO 200
47 F1=2
    CALL LBC(J,P(J))
    K=P(J)
    IF(J.LE.NT) GO TO 49
    PI(K)=-C(J)
    GO TO 50
49 PI(K)=C(J)
50 CALL PIC(K)
    GO TO 200
C    ONE ROOTED TREE
60 S=I
    IT=IL
    F2=1
    NI=0
61 S=S-1
    IT=IT-1

```

```

      IF(LS(S).EQ.LS(IT)) GO TO 61
      E=XU(J)-XL(J)
      F1=1
      K=1
62  IF(K.EQ.IT) GO TO 67
      K=K+1
      IB=LS(K)
      IF(B(K).GT.0) GO TO 63
      T=XU(IB)-X(IB)
      GO TO 64
63  T=X(IB)-XL(IB)
64  IF(T.GE.E) GO TO 62
      E=T
      LV=IB
      NI=K
      IB=P(LV)
      IF(D(IB).EQ.LV) GO TO 65
      NC=P(LV)
      GO TO 62
65  NC=Q(LV)
      GO TO 62
67  K=IL
66  IF(K.EQ.S) GO TO 68
      K=K+1
      IB=LS(K)
      IF(B(K).GT.0) GO TO 69
      T=XU(IB)-X(1B)
      GO TO 70
69  T=X(IB)-XL(IB)
70  IF(T.GE.E) GO TO 66
      E=T
      LV=IB
      NI=K
      IB=P(LV)
      IF(D(IB).EQ.LV) GO TO 71
      NC=P(LV)
      GO TO 66
68  IF(E.EQ.0) GO TO 72
      B(1)=-B(1)
DO 73 IZ=1,IT
      IB=LS(IZ)
73  X(IB)=X(IB)-B(IZ)*E
      IL=IL+1
      IF(IL.GT.S) GO TO 72
      DO 74 IZ=IL,S
      IB=LS(IZ)
74  X(IB)=X(IB)-B(IZ)*E
72  IF(NI.EQ.0) GO TO 200
      IF(NI.LE.IL-1) GO TO 75
      K=P(J)
      L=Q(J)

```

```

      IF((S.EQ.0.AND.IT.EQ.2).OR.(S.EQ.1.AND.IT.EQ.1)) GO TO 91
      CALL LBC(J,Q(J))
      GO TO 92
91    IF(U(K).EQ.K(L)) GO TO 93
      M1(U(K)
97    IF(P(M1).EQ.K) GO TO 94
      M1=P(M1)
      GO TO 95
94    M1=Q(M1)
95    IF(R(M1).EQ.D(L)) GO TO 96
      M1=R(M1)
      GO TO 97
93    U(K)=J
      GO TO 98
96    R(M1)=J
98    D(1)=J
92    PI(L)=PI(K)-C(J)
      CALL PIC(L)
      GO TO 200
75    L=P(J)
      K=Q(J)
      IF((S.EQ.0.AND.IT.EQ.2).OR.(S.EQ.1.AND.IT.EQ.1)) GO TO 191
      CALL LBC(J,P(J))
      GO TO 192
191   IF(U(K).EQ.D(L)) GO TO 193
      M1=U(K)
197   IF(P(M1).EQ.K) GO TO 194
      M1=P(M1)
      GO TO 195
194   M1=Q(M1)
195   IF(R(M1).EQ.D(L)) GO TO 196
      M1=R(M1)
      GO TO 197
193   U(K)=J
      GO TO 198
196   R(M1)=J
198   D(L)=J
192   PI(L)=PI(K)+C(J)
      CALL PIC(K)
      GO TO 200
C      OUTPUT SECTION
      100 CONTINUE
C***** BEGIN ROUTINE TO CONVERT FLOWS ON ARCS
C***** TO FLOWS ON PATHS
      DO 115 I=1,ARCS
      XB(I+1)=0
      K=P(I)
      KK=Q(I)
C***** DETERMINE WHICH CONSTRAINTS ARE ACTIVE
      CB=C(1)-PI(K)+PI(KK)

```

```

        IF(CB.GT.0.AND.XL(I).GT.0) GO TO 116
        ACT(I)=1
        XB(I)=XU(I)=X(I)
        GO TO 115
116  ACT(I)=-1
        XB(I+1)=X(I)=XL(I)
115  CONTINUE
        DO 110 I=1,N
        LBASIS(I)=0
        DO 110 J=1,N
110  BASIS(I,J)=0
106  KK=0
        I=M
        EPA=999999
        LB=0
        DO 111 K=1,ARCS
111  PATH(K)=0
103  KK=KK+1
        DO 120 J=L,N
        IF(P(J).NE.I) GO TO 120
        IF(X(J).NE.0) GO TO 101
120  CONTINUE
        GO TO 113
101  PATH(KK)=J
C***** CALCULATE LENGTHS OF PATHS IN BASIS
        LB=LB+C(J)
        IF(X(J).GE.EPS) GO TO 102
        EPS=X(J)
        JAL=J+1
102  I=Q(J)
        IF(I.EQ.1) GO TO 104
        GO TO 103
104  DO 105 I=1,KK
        J=PATH(I)
C***** SET-UP THE BASIS
        BASIS(J+1,JBL)=1
        X(J)=X(J)-EPS
105  CONTINUE
        LBASIS(JBL)=LB
        BASIS(1,JBL)=-1
C***** ASSIGN VALUES TO RHS
        XB(JBL)=EPS
        XB(1)=XB(1)+EPS
        GO TO 106
113  BASIS(1,1)=1
        DO 112 I=2,N
        IF(BASIS(I,1).GE..5) GO TO 112
        BASIS(I,1)=1
        IF(ACT(I=1).LT.0) BASIS(I,1)=-1
112  CONTINUE

```

```

RETURN
END

```

```

SUBROUTINE LBC(J,ND)
COMMON F1,F2,PI(100),D(100),U(100),R(100),NC,LV
DIMENSION P(100),Q(100),C(100)
COMMON/NETWOR/NODES,ARCS,SOURCE,P,Q,C,UC,LC,X
INTEGER F1,F2,PI,D,U,R,P,Q,C
IF(F1.EQ.1) GO TO 1
KD=D(ND)
KR=R(ND)
D(ND)=J
R(ND)=0
K=ND
IF(ND.EQ.NC) GO TO 100
GO TO 2
1  KD=D(ND)
   IF(ND.EQ.P(J)) GO TO 12
   K=P(J)
   GO TO 13
12  K=Q(J)
13  KR=R(ND)
   D(ND)=J
   R(ND)=U(K)
   U(K)=J
   IF(ND.EQ.NC) GO TO 100
   K=ND
2   L=K
   IF(L.EQ.P(KD)) GO TO 3
   K=P(KD)
   GO TO 4
3   K=Q(KD)
4   IF(U(K).EQ.KD) GO TO 5
   M=U(K)
9   IF(P(M).EQ.K) GO TO 6
   I=P(M)
   GO TO 7
6   I=Q(M)
7   IF(R(I).EQ.KD) GO TO 8
   M=R(I)
   GO TO 9
5   U(K)=KR
   GO TO 10
8   R(I)=KR
10  IF(K.EQ.NC) GO TO 11
   I=KD
   KD=D(K)
   KR=R(K)
   D(K)=I
   R(K)=U(L)
   U(L)=I
   GO TO 2

```



```

11 IF(F2.EQ.1) GO TO 100
   R(K)=U(L)
   U(L)=KD
   D(K)=DK
100 RETURN
   END

```

```

SUBROUTINE PIC(I)
COMMON F1,F2,PI(100),D(100),U(100),R(100),NG,LV
DIMENSION P(100),Q(100),C(100)
COMMON/NETWOR/NODES,ARCS,SOURCE,P,Q,C,UC,LC,X
INTEGER F1,F2,PI,D,U,R,P,Q,C
IF(U(I).EQ.0) GO TO 100
K=I
4 J=U(K)
6 IF(K.NE.P(J)) GO TO 1
  L=Q(J)
  PI(L)=PI(K)-C(J)
  GO TO 2
1 L=P(J)
  PI(L)=PI(K)+C(J)
2 IF(U(L).EQ.Q) GO TO 3
  K=L
  GO TO 4
3 IF(R(L).EQ.0 GO TO 5
  J=R(L)
  GO TO 6
5 IF(K.EQ.1) GO TO 100
  L=K
  J=D(L)
  IF(P(J).EQ.L) GO TO 7
  K=P(J)
  GO TO 3
7 K=Q(J)
  GO TO 3
100 RETURN
   END

```

```

SUBROUTINE REVISE(LENGT,UC,LC)
DIMENSION STORE(100),UPDATE(100),BINV(100,100),XB(100),
+BASIS(100,100),PRICE(100),PATH(100),LBASIS(100),
+LENGT(100),UC(100),LC(100),SUMM(100),ACT(100)
COMMON /PATH/ PATH,LENGTH,SIZE
COMMON /SOLN/ DIM,BINV,XB,BASIS,LBASIS,ACT
INTEGER DIM,SIZE,PATH,SLACK,UC
INTEGER PRICE,ACT,HEKP,CHGTST
C**** INITIALIZE DATA
IF=999999
EPS=.001
ONE=1.

```

```

        ZERO=0
        NA=DIM=1
C**** PASS PRICES DOWN
1000 CONTINUE
1001 DO 101 I=2,DIM
        K=I
        IF(BINV(1,I)) 103,101,101
101 CONTINUE
C**** PASS TO CONSTRAINED SHORTEST PATH ROUTINE
        LSTAR=-INF
        DO 200 I=2,DIM
        IF(LSTAR.LT.LBASIS(I) .AND. XB(I).GE.ZERO+EPS)
+LSTAR+LBASIS(I)
200 CONTINUE
        WRITE(6,5555) LSTAR
5555 FORMAT(' LSTAR=',15)
C**** ENTER A SLACK WHEN LONGEST PATH HAS ZERO FLOW
        DO 201 I=2,DIM
201 IF(LSTAR.LT.LBASIS(I)) GO TO 230
        GO TO 231
230 DO 232 K=2,DIM
232 IF(ABS(BINV(I,K)).GE.ZERO+EPS) GO TO 233
C**** 'ERROR'
233 SLACK=1
        NR=I
        DO 234 J=1,DIM
        STORE(J)=0
        UPDATE(J)=BINV(J,K)*ACT(K-1)
234 CONTINUE
        STORE(K)=ACT(K-1)
        KKK=K-1
        WRITE(6,6548) KKK
        GO TO 333
231 DO 202 I=1,NA
        SUMM(I)=0
202 CONTINUE
        DO 204 I=2,DIM
        IF(LBASIS(I).LT.LSTAR) GO TO 204
        DO 206 J=2,DIM
        SUMM(J=1)=SUMM(J=1)+BINV(I,J)
206 CONTINUE
204 CONTINUE
        DO 205 K=2,DIM
205 IF(ABS(BINV(1,K)).LE.ZERO+EPS .AND. SUMM(K-1)*ACT(K-1)
*.GE.ZERO+EPS) GO TO 103
        P=0
        DO 207 I=1,NA
        IF(ABS(BINV(1,I+1)).LE ZERO+EPS) GO TO 207
        Q=ABS(SUMM(I)/BINV(1,I+1))
        IF(Q.GT.P) P=Q
207 CONTINUE

```

```

      P=10*P
C      WRITE(6,2)
C2     FORMAT(' PI      SUM      PRICE')
      DO 208 I=1,NA
      PRICE(I)=1000.*(P*BINV(1,I+1)-SUMM(I))
208    CONTINUE
      LSTAR=LSTAR-1
      CALL CSPATH(PRICE,LENGT,LSTAR)
      IF(SIZE.GT.0) WRITE(6,6479) (PATH(I),I=1,SIZE)
6479   FORMAT(1X,2013)
      SUM=0
      DO 886 J=I,SIZE
      K=PATH(J)
      SUM=SUM+BINV(1,K+1)
886    CONTINUE
      IF(SUM.LT.1) GO TO 121
C****  TEST OPTIMALITY
      IF(SIZE.EQ.0) GO TO 999
      VALUE=0
      DO 210 I=1,SIZE
      J=PATH(I)
      VALUE=VALUE+SUMM(J)
210    CONTINUE
      IF(VALUE.LE.ZERO+EPS) GO TO 999
C****  CREATE AND UPDATE ENTERING VECTOR
121    DO 104 I=1,DIM
104    STORE(I)=0
      STORE(1)=-1
      DO 106 I=1,SIZE
      J=PATH(I)+1
106    STORE(J)=1
      DO 107 I=1,DIM
      UPDATE(I)=0
      DO 108 J=1,DIM
108    UPDATE(I)=UPDATE(I)+BINV(I,J)*STORE(J)
107    CONTINUE
      SLACK=0
      GO TO 109
C****  SET UP SLACK COLUMN WHEN PI(I) NEGATIVE
103    DO 105 J=1,DIM
      STORE(J)=0
      IF(J.EQ.K) STORE(J)=ACT(K-1)
      UPDATE(J)=BINV(J,K)*ACT(K-1)
105    CONTINUE
      KKK=K-1
      WRITE(6,6548) KKK
6548   FORMAT('SLACK= ',15)
      SLACK=1
C****  FIND DEPARTING VECTOR
109    NR=0
      RATMIN=INF

```

```

        DO 600 I=2,DIM
        IF(ABS(UPDATE(I)).LE.ZERO+EPS) GO TO 600
        IF(LBASIS(I)) 605,610,605
605  IF(UPDATE(I).LE.ZERO+EPS) GO TO 600
        RATIO=XB(I)/UPDATE(I)
        CHGTST=0
607  IF(RATMIN.LE.RATIO) GO TO 600
        RATMIN=RATIO
        NR=1
        HELP=CHGTST
        GO TO 600
C***** BLOCKING VARIABLE COULD BE SLACK
610  IF(UPDATE(I)) 611,600,615
615  RATIO=XB(I)/UPDATE(I)
        CHGTST=0
        GO TO 607
600  CONTINUE
        IF(HELP.EQ.0) GO TO 333
C***** SLACK VARIABLE IS BLOCKING -EXCHANGE CONSTRAINTS
        DO 620 J=1,DIM
        BINV(NR,J)=-BINV(NR,J)
620  GONTINUE
        XB(NR)=-XB(NR)+(UC(NR-1)-LC(NR-1))
        UPDATE(NR)=-UPDATE(NR)
        ACT(NR-1)=-ACT(NR-1)
C***** PIVOT
        333  WRITE(6,4456)NR
        4456  FORMAT('PIVOT ROW =',I3)
630  PIVELE=UPDATE(NR)
        DO 112 J=1,DIM
        112  BINV(NR,J)=BINV(NR,J)/PIVELE
        XB(NR)=XB(NR)/PIVELE
        DO 114 I=1,DIM
        IF(I.EQ.NR) GO TO 114
        XB(I)=XB(I)+UPDATE(I)*XB(NR)
        DO 116 J=1,DIM
        116  BINV(I,J)=BINV(I,J)+UPDATE(I)*BINV(NR,J)
        114  CONTINUE
C***** STORE THE PATH VECTOR
        LBASIS(NR)=0
        DO 118 I=1,DIM
        BASIS(I,NR)=STORE(I)
        IF(SLACK.EQ.1.OR.I.EQ.1) GO TO 118
        LBASIS(NR)=LBASIS(NR)+LENGT(I-1)*STORE(I)
        118  CONTINUE
        GO TO 1000
999  RETURN
        END

SUBROUTINE MATINV(N)

```

```

      DIMENSION BASIS(100,100),A(100,200),BINV(100,100)
      DIMENSION LBASIS(100),ACT(100)
      DIMENSION XB(100)
      COMMON D,OUT
      COMMON/SOLN/DIM,BINV,XB,BASIS,LBASIS,ACT
C***** AUGMENT BASIS WITH INDENITY MATRIX
      EPS=.0001
      OUT=0.
      M=2*N
      DO 100 I=1,N
      DO 95 J=1,M
      IF(J.GT.N) GO TO 80
      A(I,J) = BASIS(I,J)
      GO TO 95
80    A(I,J)=0
      IF((J-N).NE.I) GO TO 95
      A(I,J)=1
95    CONTINUE
100   CONTINUE
C***** BEGIN ELLIMINATION
C***** DETERMINANT IS D
      D=1.
      DO 200 K=1,N
      D=d*A(K,K)
C***** CHECK FOR ZERO PIVOT ELEMENT
      IF(ABS(A(K,K)).GT.EPS) GO TO 210
      WRITE(6,102)
102   FORMAT(' ZERO PIVOT ELEMENT- MAYBE ERROR IN BASIS')
      OUT=1
      GO TO 999
C***** DIVIDE ROW BY PIVOT ELEMENT
210   KK=K+1
      DO 215 J=KK,M
      A(K,J)=A(K,J)/A(K,K)
215   CONTINUE
      A(K,K)=1.
C***** SWEEP OUT K(TH) COLUMN
      DO 220 I=1,N
      IF(I.EQ.K.OR.ABS(A(I,K)).LT.EPS) GO TO 220
      DO 225 J=KK,M
      A(I,J)=A(I,J)-A(I,K)*A(K,J)
225   CONTINUE
      A(K,I)=0
220   CONTINUE
200   CONTINUE
      DO 300 I=1,N
      DO 301 J=1,N
      BINV(I,J)=A(I,J+N)
301   CONTINUE
300   CONTINUE
999   RETURN

```

END

```

SUBROUTINE PRINT
  DIMENSION BINV(100,100),XB(100),BASIS(100,100),LBASIS(100)
  DIMENSION ACT(100)
  INTEGER DIM,ACT
  COMMON/SOLN/ DIM,BINV,XB,BASIS,LBASIS,ACT
  WRITE(6,120)
120  FORMAT(' BASIS MATRIX')
  WRITE(6,106)((BASIS(I,J),J=1,DIM),I=1,DIM)
106  FORMAT(10F6.2)
  WRITE(6,121)
121  FORMAT(' BASIS INVERSE')
  WRITE(6,106)((BINV(I,J),J=1,DIM),I=1,DIM)
  RETURN
END

```

```

SUBROUTINE CSPATH (CARC,IARC,LSTAR)
  DIMENSION LSWTCH(100),LABEL(100,10,3),LISTB(100),LISTE(100)
  DIMENSION LNUMB(100),IARC(100),PATH(100),LEXIT(100)
  INTEGER CARC(100), ARCS,SAVE(100),SAVE1(100)
  INTEGER DIST,SIZE,PATH,CTEST,ENTER,EXIT
  COMMON/NETWORK/NODES,ARCS,SOURCE,LISTB,LISTE
  COMMON/PATH/PATH,LENGTH,SIZE
C***** CONSTRAINED SHORTEST PATH ALGORITHM TO GENERATE
C***** COLUMNS TO ENTER BASIS
C***** PROCEDURE USES REVERSE NUMBERING OF NETWORK
C***** NODE 1 IS THE SINK
C***** LAST NODE IS THE SOURCE
  INF=99999
  DO 10 I=1,NODES
    LEXIT(I)=0
    LSWTCH(I)=0
    LNUMB(I)=0
    SAVE(I)=0
    SAVE1(I)=0
  10 CONTINUE
C***** INITIALIZATION OF LABEL FOR SINK
  MM=1
  LSWTCH(1)=1
  LABEL(1,1,1)=0
  LABEL(1,1,2)=0
  LABEL(1,1,3)=INF
  LNUMB(1)=1
C***** SEARCH FOR NODE THAT CAN BE LABELED
95  DO 200 I=1,NODES
    IF(LSWTCH(I).EQ.1.OR.LSWTCH(I).EQ.INF) GO TO 200
    K=1
C ***** SEARCH ARCS
    DO 100 J=1,ARCS

```

```

        IF(LISTB(J).NE.I) GO TO 100
        JJ=LISTE(J)
        IF(LSWTCH(JJ).EQ.0) GO TO 200
        IF(LSWTCH(JJ).EQ.INF) GO TO 100
C ***** NODE HAS BEEN FOUND
        SAVE(K)=JJ
        SAVE1(K)=J
        K=K+1
100    CONTINUE
        IF(K.NE.1) GO TO 101
        WRITE(6,102) I
102    FORMAT(' LABEL CAN NOT BE CREATED BECAUSE LENGTH',/,
1' IS GREATER THAN LSTAR FOR ALL NODES CONNECTED TO
2NODE',I5)
        LSWTCH(I)=INF
        IHELP=I
        GO TO 200
101    KK=K+1
C ***** BEGIN LABELING
        DO 400 L=1, KK
        NN=SAVE1(L)
        LL=SAVE(L)
        LLL=LNUMB(LL)
        DO 300 M=1, LLL
C ***** IS THERE MORE THAN ONE NODE FROM
C ***** WHICH PRESENT NODE CAN BE LABELED
        IF(LSWTCH(I).NE.0) GO TO 250
        IF(LARC(NN)+LABEL(LL,M,1).GT.LSTAR) GO TO 300
        LNUMB(I)=LNUMB(I)+1
        MMM=LNUMB(I)
        LABEL(I,MMM,1)=LARC(NN)+LABEL(LL,M,1)
        LABEL(I,MMM,2)=LARC(NN)+LABEL(LL,M,2)
        LABEL(I,MMM,3)=NN
        GO TO 300
250    III=LNUMB(I)
        ENTER=0
        EXIT=0
        DO 260 II=1, III
260    LEXIT(II)=0
        DO 275 II=1, II
C ***** IF MORE THAN ONE NEW LABEL POSSIBLE
C ***** DONT ADD IF LENGTH AND COST ARE GREATER
C ***** THAN ANY OF THE PRESENT VALUES
        IF(LARC(NN)+LABEL(LL,M,1).GT.LSTAR) GO TO 300
        IF(LARC(NN)+LABEL(LL,M,1).GE.LABEL(I,II,1).AND.CARC(NN)
        ++LABEL(LL,M,2).GE.LABEL(I,II,2)) GO TO 300
        IF(LARC(NN)+LABEL(LL,M,1).GT.LABEL(I,II,1).OR.CARC(NN)
        ++LABEL(LL,M,2).GT.LABEL(I,II,2)) GO TO 270
C ***** WE CAN DELETE A LABEL
        EXIT=1

```

```

        LEXIT(II)=1
270  ENTER=1
275  CONTINUE
C ***** CAN WE CHANGE LABEL LIST
        IF(ENTER.EQ.0) GO TO 300
        IF (ENTER.EQ.1.AND.EXIT.EQ.0) GO TO 281
C ***** UP DATE LABEL LIST
        LNUMB(I)=0
        DO 280 II=1,III
        IF(LEXIT(II).EQ.1) GO TO 280
        LNUMB(I)=LNUMB(I)+1
        JJ=NUMB(I)
        LABEL(I,JJ,1)=LABEL(I,II,1)
        LABEL(I,JJ,2)=LABEL(I,II,2)
        LABEL(I,JJ,3)=LABEL(I,II,3)
280  CONTINUE
281  LNUMB(I)=LNUMB(I)+1
        JJ=LNUMB(I)
        LABEL(I,JJ,1)=LARC(NN)+LABEL(LL,M,1)
        LABEL(I,JJ,2)=LARC(NN)+LABEL(LL,M,2)
        LABEL(I,JJ,3)=NN
300  CONTINUE
        IF(LNUMB(I).EQ.0) GO TO 400
        LSWTCH(I)=1
400  CONTINUE
        IF(LNUMB(I).NE.0) GO TO 200
        LSWTCH(I)=INF
200  CONTINUE
        SIZE=0
        IF(IHELP.EQ.NODES) GO TO 900
        CTEST=INF
        IIII=LNUMB(NODES)
        IF(IIII.LT.1) GO TO 95
C ***** PROCEDURE TO FIND PATH TO PASS
C ***** TO REVISED SIMPLEX SUBROUTINE
        DO 500 II=1,IIII
        IF(LABEL(NODES,II,1).GT.(LSTAR)) GO TO 500
        IF(LABEL(NODES,II,2).GE.CTEST) GO TO 500
        CTEST=LABEL(NODES,II,2)
        M=II
500  CONTINUE
        IF(CTEST.EQ.INF) GO TO 604
        GO TO 699
604  WRITE(6,605)
605  FORMAT(' NO PATH OF SHORTER LENGTH HAS BEEN FOUND')
        GO TO 900
699  I=NODES
700  JJ=LABEL(I,M,3)
        DIST=LABEL(I,M,1)=LARC(JJ)
        SIZE=SIZE+1

```



```
      PATH(SIZE)=JJ
      J=LISTE(JJ)
      IF(J.EQ.1) GO TO 900
      K=LNUMB(J)
      DO 705 N=1,K
      M=N
      IF(LABEL(J,N,1).EQ.DIST) GO TO 710
705  CONTINUE
      WRITE(6,800)
800  FORMAT(' MISTAKE IN THE LABELING PROCEDURE')
      GO TO 900
710  I=J
      GO TO 700
900  RETURN
      END
```

- - - T H E E N D - - -

## APPENDIX B

### INPUT TO AND OUTPUT FROM THE COMPUTER PROGRAM

In Appendix A of this thesis is a copy of the computer code, written in FORTRAN IV, for the complete solution strategic transportation problem as described in Chapter I. This program includes the algorithm, for the least time transportation problem in Chapter II, the network generation scheme of Chapter III, and the algorithm to solve the ship scheduling subproblem. The only required input to the computer program is the information necessary for the solution of the Base-to-Port transportation problem (i.e. least time transportation problem). The general data required by the program is:

- (1) the number of bases
- (2) the number of ports
- (3) the number of ships
- (4) the base to port travel time matrix, BP
- (5) the ship to port travel time matrix, SP
- (6) the port to objective area travel time vector, PO
- (7) the vector of available ship loads of supplies at the bases
- (8) the vector required ship loads of supplies at the ports.

The output of the program consists of three sections: a listing of the input data, certain intermediate results, and the optimal solution. The listing of data is a table of all the input information that can be used for verification of the input data. Among

the intermediate results displayed are: (1) the maximal flow through the network; (2) a list of the path length and the amount of flow on each of these paths which constitute the maximal flow solution; (3) an indicator of which constraint is binding for each arc in the network - a "+1" refers to a type A constraint and a "-1" refers to a type B constraint; (4) the basis and basis inverse matrices from the maximal flow solution; and (5) a series of values which are concerned with the reduction of the flow on the longest path (or paths) through the network. LSTAR is the length of the longest path carrying flow. SLACK is the number of the slack variable entering when applicable. When a path enters the arc numbers will be printed, e.g., "2 5 8". PIVOT ROW is the row in which the variable (path or slack) entered.<sup>†</sup> The optimal solution is displayed in tableau form, with the BASIS indicating which column vector a path was in the basis matrix of the optimal solution to the ship scheduling problem, FLOW indicates the flow along each path, and LENGTH indicates the length of each path. PATH is the portion of the table which is of most interest, since it will be the schedule for the ships. Each path corresponds to the travel of a ship through the network, with arc of the path being a portion of the travel (i.e., one arc might indicate movement from some point at sea to a port). If path 3, corresponding to the third ship being scheduled, uses arc 8, 10, 12 for instance, then these arcs will determine which port ship 3 must travel to, when compared with the input network. Thus the optimal (i.e., minimal closure) ship schedule is displayed in this table. The

---

<sup>†</sup>Note that the first row of BINV contains the 's. Rows 2 through m+1 are associated with the arcs.

closure time of the ship scheduling subproblem is the maximum LENGTH of the paths in the optimal solution, since length actually corresponds to a travel time. Due to the method of construction of the ship scheduling network the closure time of the ship scheduling subproblem is the same as that for the overall problem.

The following is an example output from the computer program.

#### INTERMEDIATE TABLEAUS FOR LIT

##### TABLEAU FROM NORTHWEST CORNER RULE

1.00	.00
1.00	3.00

##### OTHER TABLEAUS

.00	1.00
2.00	2.00

##### OTHER TABLEAUS

.00	1.00
2.00	2.00

##### OPTIMAL SOLUTION TO LIT

##### FINAL TABLEAU

0	1
2	2

##### SUPPLY MATRIX

0	0
0	1
2	0
0	2

NUMBER OF BASES: 2  
 NUMBER OF PORTS: 2  
 NUMBER OF SHIPS: 5  
 NUMBER OF NODES: 23  
 NUMBER OF ARCS: 40  
 SOURCE IS: 23

##### NODES, ARCS, SOURCE? ARC DATA?

NUMBER OF NODES IN THE NETWORK IS 23  
 NUMBER OF ARCS IN THE NETWORK IS 37  
 THE FOLLOWING DATA IS USED:

BEGINNING NODE	ENDING NODE	LENGTH	UPPER	LOWER	CAPACITY
2	1	0	7		2
3	1	0	8		3
6	10	4	2		0
7	10	4	2		0
8	10		2		0
9	10	4	2		0
4	5	1	2		0
5	6	1	2		0
6	7	1	2		0
7	8	1	2		0
8	9	1	2		0
10	2	0	2		0
12	16	3	1		0
12	16	3	1		0
13	16	3	1		0
14	17	3	3		0
15	17	3	3		0
11	12	1	3		0
12	13	1	3		0
13	14	1	3		0
14	15	1	3		0
16	3	0	1		0
17	3	0	3		0
18	9	6	1		0
18	12	2	1		0
19	6	3	1		0
19	11	1	1		0
20	4	1	1		0
20	15	5	1		0
21	7	4	1		0
21	13	3	1		0
22	6	3	1		0
22	15	5	1		0
23	18	0	1		0
23	19	0	1		0
23	20	0	1		0
23	21	0	1		0
23	22	0	1		0

MAX FLOW = 5.00

ARC	ACTIVE TYPE	PATH LENGTH	RIGHT HAND SIDE VECTOR
1	-1	0	.00
2	-1	0	.00
3	1	0	.00
4	1	0	2.00
5	1	0	2.00

ARC	ACTIVE TYPE	PATH LENGTH	RIGHT HAND SIDE VECTOR
6	1	0	2.00
7	1	0	1.00
8	1	0	1.00
9	1	0	2.00
10	1	0	2.00
11	1	0	2.00
12	1	0	.00
13	1	0	.00
14	1	0	1.00
15	1	0	1.00
16	1	0	1.00
17	1	0	2.00
18	1	0	2.00
19	1	0	1.00
20	1	0	3.00
21	1	0	.00
22	1	0	1.00
23	1	0	1.00
24	1	0	.00
25	1	0	1.00
26	1	0	.00
27	1	0	.00
28	1	0	1.00
29	1	0	1.00
30	1	0	.00
31	1	0	.00
32	1	0	1.00
33	1	5	1.00
34	1	7	1.00
35	1	7	1.00
36	1	7	1.00
37	1	7	1.00

THE DETERMINANT OF BASIS IS 1.00

LSTAR = 7

34 26 17 13 21 2

PIVOT ROW= 14

LSTAR= 7

33 24 18 14 21 2

PIVOT ROW= 22

LSTAR= 7

33 24 13 21 2









[illegible]



.00	.00	.00	1.00	.00	.00	.00	.00	.00	.00
.00	-1.00	.00	.00	.00	.00	.00	.00	.00	.00
.00	.00	.00	.00	.00	.00	.00	.00	.00	.00
.00	.00	.00	.00	.00	.00	.00	.00	.00	.00
.00	.00	1.00	.00	.00	.00	.00	.00	.00	.00
.00	.00	.00	.00	.00	.00	.00	.00	.00	.00
.00	.00	.00	.00	.00	.00	.00	.00	.00	.00
.00	.00	.00	.00	.00	.00	.00	.00	.00	.00
.00	.00	.00	.00	.00	.00	.00	.00	.00	.00
.00	1.00	.00	.00	.00	.00	.00	.00	.00	.00
.00	.00	.00	.00	.00	.00	.00	.00	.00	.00
.00	.00	.00	.00	.00	.00	.00	.00	.00	.00
.00	.00	.00	.00	.00	.00	.00	.00	.00	.00
1.00	.00	.00	.00	.00	.00	1.00	.00	.00	.00
.00	.00	.00	.00	.00	.00	.00	.00	.00	.00
.00	.0000	.00	.00	.00	.00	.00	.00	.00	.00
.00	.00	.00	.00	.00	.00	.00	.00	.00	1.00
.00	.00	.00	.00	.00	-1.00	.00	.00	.00	.00
.00	.00	.00	.00	.00	.00	.00	.00	.00	.00
.00	.00	.00	.00	.00	.00	.00	.00	.00	.00
.00	.00	.00	.00	.00	.00	.00	.00	1.00	.00
.00	.00	.00	.00	.00	.00	.00	.00	.00	.00
.00	.00	.00	.00	.00	.00	.00	1.00	.00	.00
.00	.00	.00	.00	.00	-1.00	.00	.00	.00	.00
.00	.00	.00	.00	.00	.00	.00	1.00	.00	.00
.00	.00	.00	.00	.00	.00	.00	.00	.00	.00
.00	.00	.00	.00	.00	.00	.00	.00	.00	.00
.00	.00	.00	.00	.00	.00	.00	.00	.00	.00
.00	.00	.00	-1.00	.00	.00	.00	.00	.00	.00
.00	.00	.00	.00	.00	.00	.00	.00	.00	.00
.00	.00	.00	.00	.00	1.00	1.00	.00	.00	.00
.00	.00	.00	.00	.00	.00	.00	.00	.00	.00
.00	.00	.00	.00	.00	.00	.00	.00	.00	.00
.00	.00	.00	.00	.00	.00	.00	.00	.00	.00
.00	.00	.00	.00	.00	.00	.00	.00	.00	.00
.00	.00	.00	.00	.00	1.00	.00	.00	.00	.00
.00	.00	.00	.00	.00	.00	.00	.00	.00	.00
.00	.00	.00	.00	.00	.00	.00	.00	.00	.00
.00	.00	.00	.00	.00	.00	.00	.00	.00	.00
.00	.00	.00	.00	1.00	.00	.00	.00	.00	.00
.00	.00	.00	.00	.00	.00	.00	.00	.00	.00
.00	.00	.00	.00	.00	.00	.00	.00	.00	.00
.00	.00	.00	.00	.00	.00	.00	.00	.00	.00
.00	.00	.00	1.00	.00	.00	.00	.00	.00	.00

## OPTIMAL SOLUTION

BASIS	FLOW	LENGTH	PATH
2	.00	0	1
3	.00	0	2
4	.00	0	3
5	2.00	0	4
6	2.00	0	5
7	2.00	0	6
8	1.00	0	7

[illegible]

## REFERENCES

1. Hinkle, Robert G., Min-Max Path Flow in Directed Networks, Ph.D. Dissertation, Georgia Institute of Technology.
2. Dantzig, George B., Linear Programming and Extensions, Princeton University Press, Princeton, New Jersey, 1963.
3. Taha, Hamdy, D., Operations Research: An Introduction, Macmillan, New York, New York, 1971.
4. Ford, L. R. and D. R. Fulkerson, Flows in Networks, Princeton University Press, Princeton, New Jersey, 1962.
5. Robers, Philip and Adi Ben-Israel, Interval Programming, I & EC Process Design and Development, 8, No. 4., 496-501, October, 1969.
6. Berry, Ronald C. and Paul A. Jensen, A Constrained Shortest Path Algorithm, 39th National ORSA Meeting, Dallas, Texas, May 5-7, 1971.
7. Agin, Norman, Optimum Seeking with Branch and Bound, Management Science, Vol. 13, No. 4, Dec. 1966.

## VITA

Peter Dean Keith was born in Hinsdale, Illinois on July 30, 1949, the son of Kenneth Burton Keith and Gloria Johnson Keith. In 1967, after graduating from St. Procopius Academy in Lisle, Illinois, he entered the University of Michigan. He graduated from the University of Michigan with a Bachelor of Science in Industrial Engineering in December, 1971. After a year of work and travel, he entered the Graduate School of Georgia Institute of Technology in January, 1973. He is presently planning to continue his education at Northwestern University, Evanston, Illinois in the area of urban systems.