OPTIMAL ROUTING OF LARGE SCALE MARINE SEISMIC MAPPING OPERATIONS

By

THOMAS CHIH-HSIUNG CHEN

Bachelor of Science
Tunghai University
Taichung, Taiwan
1968

Master of Science
University of Pittsburgh
Pittsburgh, Pennsylvania
1971

Submitted to the Faculty of the Graduate College
of the Oklahoma State University
in partial fulfillment of the requirements
for the Degree of
DOCTOR OF PHILOSOPHY
July, 1976

OPTIMAL ROUTING OF LARGE SCALE MARINE SEISMIC

MAPPING OPERATIONS

Thesis Approved:

M. Palmer Terrell

Thesis Adviser

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

Dean of the Graduate College

964122

PREFACE

This study is concerned with the optimal routing of large scale marine seismic mapping operations. The purpose of marine seismic exploration is the detection of oil reserves beneath the floor of the ocean. Included in this operation is the scheduling of the geophysical recording ships which collect seismic information at a particular prospect. Computational algorithms developed in this study can be used to schedule a single-ship through single-prospect problem, a single-ship through multi-prospects problem, and a multi-ships through multi-prospects problem. The algorithms developed are very effective with respect to core storage requirements and computation times.

Although the primary result of this dissertation will be the reduction of the managerial decision making difficulties involved in scheduling geophysical ships, the result can also be extended to the solution of constrained traveling salesman problem and the machine sequence scheduling problem.

The author wishes to express his appreciation to his major adviser, Dr. M. Palmer Terrell, for his guidance and encouragement throughout this research and for his assistance in the preparation of the final manuscript. Special appreciation is also expressed to Dr. Syed Z. Shariq whose assistance and suggestions to this study are invaluable. Appreciation is also expressed to committee members, Dr. Donald W. Grace, Dr. James E. Shamblin, and Dr. Wayne C. Turner, for their guidance and generous contribution to this study and to my education at Oklahoma

State University. Appreciation is also expressed to Miss Velda D. Davis and Mrs. Margaret Estes for their excellent typing.

Special appreciation is expressed to my parents, Mr. and Mrs. Ying-Ho Chen, for instilling within me the importance of an education and for their understanding and confidence in me as a student and as a son.

Finally, sincere appreciation is given to my wife, Pearl, for her encouragement, understanding, and patience throughout these years of graduate study.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Figure Page

CHAPTER I

INTRODUCTION

Statement of the Problem

In recent years, offshore exploration and production of oil and gas has increased sharply due to the scarcity of energy resources on the continent. There are three major geophysical methods for oil and gas exploration, i.e., gravity, magnetic, and seismic [34]. Among them, the seismic reflection method is the most inexpensive and effective. It is commonly used in detecting the oil and gas reserves beneath an ocean floor.

To collect data by the seismic reflection method, several shot points beneath the ocean floor are explored along the prospects survey lines, and the magnitudes of the noise reflections are recorded by a series of sensitive devices. These sensitive devices are located at equal intervals on a seismic marine cable which is towed by a ship. This cable, usually from one-half of a mile to two miles in length, must be aligned with the ship's movement as data is collected.

A "prospect" is a geographical location to be searched for oil or gas reserves. A prospect consists of a configuration of N straight lines which indicates the paths that a ship will cover while collecting the seismic data. The number of lines N at a given prospect might be anywhere from two to two dozen, with individual line length ranging from five miles to one hundred miles. To collect the required data by a single-ship through single-prospect, a ship must leave a known port P_1 and travel to the prospect, traverse each of the N lines one-way collecting data, then return to a known port P_2 (Figure 1). The ship should be routed through such a path-configuration so as to collect the required data efficiently. For a large geophysical company, it is sometimes necessary to route a single-ship through multi-prospects or multi-ships through multi-prospects. A problem of this type is a large scale seismic mapping problem and the routing procedure is more complicated. Since the number of feasible paths that could be selected (for a single-ship through single-prospect with N lines there are $2^N N!$ possible paths) is extremely large and the maintenance of a geophysical crew is very high, it is economically desirable that an effective technique for determining optimal ship routings for seismic mapping be developed.

Research Objectives

The objectives of this research are two fold. The first objective is to develop algorithms that can be used to determine the minimum cost route for the single-ship through single-prospect problem, and that improve upon a previous algorithm developed by Willard [66] in the areas of execution time and storage requirements. The second objective is to extend the best of the above algorithms to develop an algorithm that can solve the large scale seismic mapping problem (e.g., the multi-ships, multi-prospects operation). The primary result of this research will be the reduction of the managerial decision making difficulties involved in scheduling geophysical ships. In addition, the results from this research can be extended to the solution of constrained traveling salesman problems and machine sequence scheduling problems.

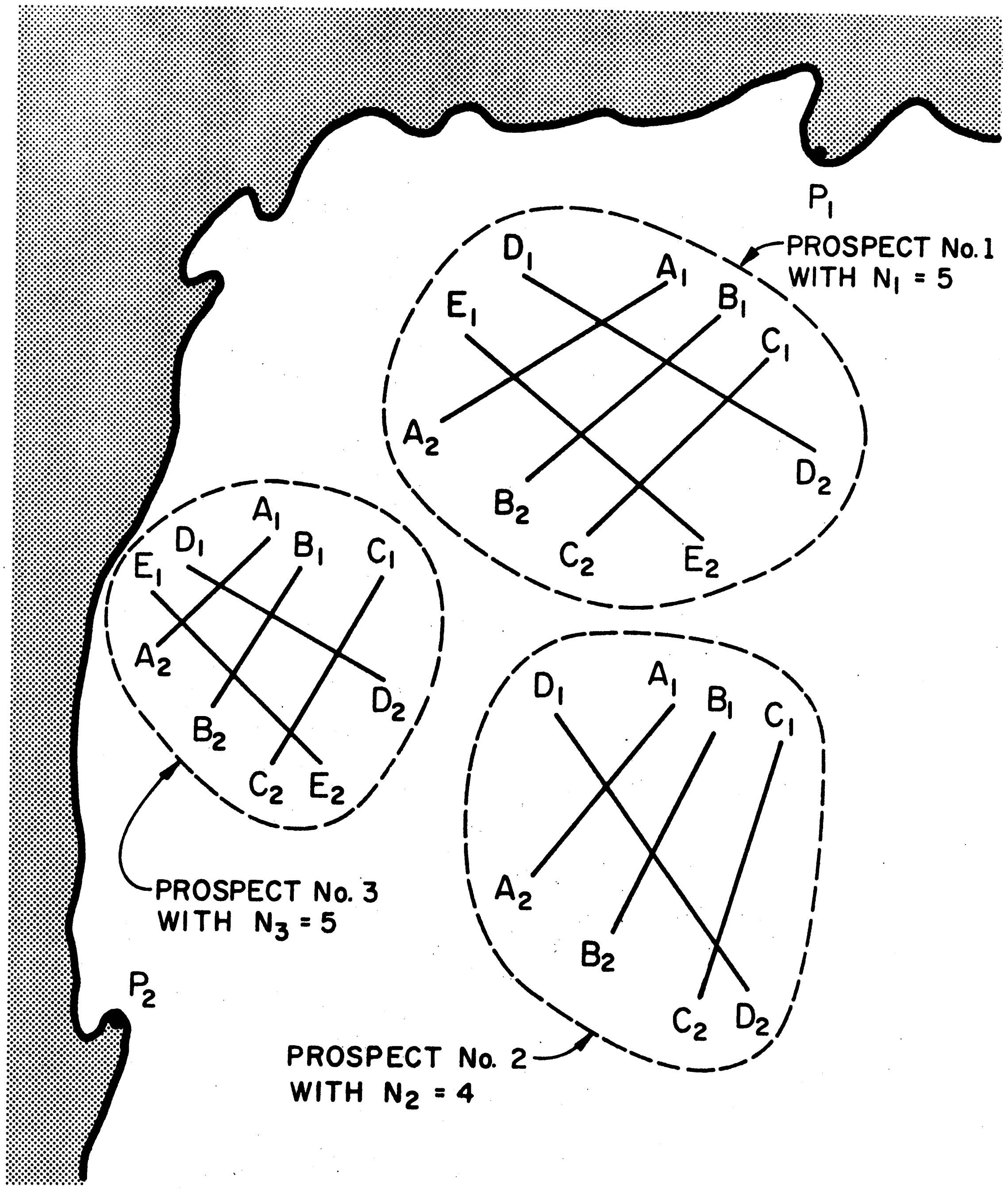


Figure 1. An Example of Seismic Prospect Configuration: Three Prospects

Research Procedure and Methodology

The development of algorithms in this research is done in two phases:

<u>Phase I</u>

1. Study the applicability of the dynamic programming approach to the problem. Develop an improved algorithm for solving the single-ship through single-prospect problem by this technique.
2. Study the applicability of the branch and bound approach to the problem. Develop an algorithm for solving the single-ship through single-prospect problem by this technique.
3. Study the applicability of the graph theory approach to the problem. Develop an algorithm for solving the single-ship through single-prospect problem by this technique.
4. Conduct the performance comparisons of execution times and storage requirements for above developed algorithms for different prospect path-configurations with various number of lines.

<u>Phase II</u>

1. Using the best approach from Phase I, possibly in some required combination, develop a new algorithm for solving the single-ship through multi-prospects and the multi-ships through multi-prospects problem.
2. Test the performance of the algorithm considering execution times and storage requirements for different prospect

path-configurations with various number of lines.

3. Document all computer algorithms and develop a user's guide.

In the dynamic programming approach, a recursive search dynamic programming method is studied and applied to the problem. An algorithm is developed based on this approach, and the effectiveness of the algorithm is then compared to the effectiveness of the conventional dynamic programming method considering the storage requirements and computation times.

In the branch and bound approach, two methods, Little's sequential tour-building method and Eastman's subtour elimination method are studied and applied to the problem. Two algorithms are developed based on these two methods. A modification of Little's sequential tour-building method is selected finally for a further research in the single-ship through multi-prospects and the multi-ships through multi-prospects problem.

In the graph theoretic approach, the shortest spanning tree and shortest Hamiltonian chain concepts are studied and applied to the problem. An algorithm is developed based on the combination of these two concepts. The application of this algorithm to a machine sequence scheduling problem is also studied.

CHAPTER II

LITERATURE REVIEW

After an extensive literature search, it was determined that very little has been published pertinent to this specific problem. However, there is a similarity between the problem in this research and the classical traveling salesman problem. In the traveling salesman problem it is assumed that there are N towns with known distances between any two of them. A salesman wants to start from a given town, visit each town once, and then return to his starting point. The objective is to minimize his total traveling time. A seismic mapping problem with an N line seismic path-configuration can be thought of as a traveling salesman problem with $2N + 2$ cities and the restriction that cities be visited in specified pairs. The unconstrained traveling salesman problem has been treated by a number of persons using a variety of techniques.

One of the earliest investigations was made by Dantzig, Fulkerson, and Johnson [14] in 1954. Their papers published in 1954 and 1958 outline a linear programming approach to the problem. Their approach starts with an arbitrary solution, then employs the standard Simplex method to improve the basis. A link in the basis is replaced by a new link in each iteration. Since a link which has been removed can be reintroduced at a later iteration, the approach is highly inefficient. Because of the additional constraints that would need to be imposed, a linear programming formulation of the problem would be very large.

In 1956, Flood [22] related the traveling salesman problem to personnel-assignment problems. The traveling salesman problem is different from the assignment problem only in that the allowable permutation of N persons and M jobs must be a cycle. In the paper, Flood illustrated how the assignment method could be used effectively in the initial preparation of a traveling salesman problem for subsequent computations. Some techniques that are useful in seeking good approximate solutions are given. Also, in 1956, Kruskal [38] pointed out a possible relation between the traveling salesman problem and shortest spanning tree problem. At about the same time, Barachet [4] reported a graph theoretical approach for the solution of traveling salesman problem. Both concepts have been extended by many authors [10, 32, 33, 47] in later years.

Croes [12] developed a tour to tour improvement approach in 1958 to solve the traveling salesman problem. Croes' solution generation scheme is a rule for finding a better tour that is a neighbor of the present tour. The results of this procedure are approximate and the procedure is inefficient for a problem with a large number of cities.

Eastman [19] originated the branch and bound approach. Eastman's method capitalizes on the fact that every solution to the traveling salesman problem is also a feasible solution to the corresponding assignment problem. The optimal solution to the assignment problem is therefore a lower bound to the solution of the traveling salesman problem; and if the solution is cyclic, it is also the optimal solution to the traveling salesman problem. If the solution is not cyclic, one or more subtours must exist. Eastman chose the subtour having the minimum number of arcs and created a set of new problems by eliminating arcs from the subtour, one at a time. In later years, Bellmore and Malone [8] extended Eastman's subtour elimination method. Miller, Tucker, and Zemlin [44] formulated the traveling salesman problem as an integer programming problem. Using this technique an N-city problem required $N^2 + N$ constraints and N^2 variables. The authors concluded that the integer programming procedure was highly inefficient.

Bellman [5] and Held and Karp [31] independently applied the dynamic programming method to obtain the optimal route for the traveling salesman problem. Using this approach the traveling salesman was formulated as a multi-stage decision problem. The optimal path segments obtained from a particular stage are retained and used in obtaining the optimal segmental routes in subsequent stages. The drawback of this method is the large computer storage requirement for solving a problem with a large number of cities.

Subsequently, Little, Murty, Sweeney, and Karel [42] developed a branch and bound algorithm for the traveling salesman problem. This algorithm divides paths into two categories; paths that contain a directional link connecting two particular cities and all remaining paths that exclude the selected link. At every stage where this separation of paths or "branching" occurs, a lower bound is calculated for each of the sets of paths within each of the above categories. At each branching stage, the directional link is selected in such a manner that the lower bound for the set of paths not containing the link in question will be as large as possible. The optimal route is determined once a circuit is found where the total distance required to be traveled is smaller than the lower bound of each of the other path segments.

Because of the obstacles faced by optimal seeking procedures for the traveling salesman problem with a large number of cities, Karg and Thompson [37] approached the problem by heuristic procedures. The method begins with a randomly selected pair of cities, constituting a tour of length 2. Then a third city is inserted in order to minimize the resulting three-city tour; then a fourth city is inserted, and so on, until a complete tour has been constructed. The heuristic approach has been extended by Lin and Kernighan [41].

Obruca [47] observed that the majority of lines appearing in a shortest spanning tree for any network also correspond to those in the solution of a traveling salesman problem with the same network. The technique developed by Obruca is to manipulate the tree by means of deletions and additions of lines into a chain and hence obtain a feasible solution. In applying this procedure, Obruca points out that among 460 sets of randomly generated cost matrices with N varying from 5 to 11 cities, only 50% of the solutions were identical with the optimal.

In 1970, Christofides [10] combined the shortest spanning tree and branch and bound algorithm to develop two new algorithms to solve the traveling salesman problem. The first algorithm is based on a decision-tree search with lower bound used to limit the search. The second algorithm is a fast iterative procedure based on simple transformations of the cost matrix of the graph, ensuring at each step that the relative cost of all Hamiltonian chains stay unchanged. At about the same time, Held and Karp [32] approached the symmetric traveling salesman problem by the 1-tree concept, which is a slight variant of spanning trees. A 1-tree is a tree together with an additional vertex connected to the tree by two edges.

Since the traveling salesman problem was studied by Dantzig, Fulkerson, and Johnson [14], most of the subsequent algorithm were developed to deal only with the single traveling salesman problem. Svestka and Huckfeldt [61] investigated an M-salesman traveling salesman problem in 1973. The multiple salesman traveling salesman problem can be defined as : Given M salesmen and N cities, find M sorties such that every city (except the home city) is visited exactly once by exactly one salesman, so that the total distance traveled by all salesmen is minimum. The authors applied the Eastman's subtour elimination algorithm to solve an M-salesman traveling salesman problem. Some computation experience has been reported by the authors.

Besides the traveling salesman problem, the recursive search method of dynamic programming has also been studied in this research and the review is presented in the following. Most of the work on allocation problems with integer solutions has been accomplished with dynamic programming. However, the fact that the computer memory requirement increases exponentially with the increasing size of the problem has limited its usage. Williams [67], in solving an allocation problem, has proposed a recursive search method of dynamic programming to overcome the limitations of conventional dynamic programming. With this technique, only a limited number of states and decision variables in each stage require investigating, so that computational times and computer memory requirements are significantly reduced.

The most significant contribution to this research of marine seismic mapping operation problem was that of Willard [66]. Willard used a dynamic programming method to approach the single-ship through single-prospect problem. Although this attack is effective in execution time, the algorithm was faced with a storage problem for an arbitrarily large number of lines. The algorithm developed by the author using the FORTRAN IV computer language, required a computer memory capacity of approximately 250 K bytes for a ten line path-configuration. The dynamic programming approach of Willard to the proposed problem is a starting point and provides a spring board for the research reported in the following chapters.

CHAPTER III

ALGORITHM FOR THE SINGLE-SHIP THROUGH SINGLE-PROSPECT PROBLEM USING A RECURSIVE SEARCH DYNAMIC PROGRAMMING APPROACH

Willard [66] used the dynamic programming method to solve the single-ship through single-prospect problem. The algorithm developed by Willard was faced with a storage problem for an arbitrarily large number of lines. The algorithm, using the FORTRAN IV language, required a computer memory capacity of approximately 250 K bytes for a ten line configuration. This research investigates a possible method that can reduce the memory storage with some trade-off of computation time, while the concepts of the dynamic programming approach are still maintained.

After reviewing various search techniques, it is believed that the recursive search dynamic programming method can considerably reduce the computer storage, but would require some additional computation time. Basically, the recursive search technique, starting with a feasible solution, searches over each of the recursive relationships until an optimum solution is reached. With this technique, limited numbers of state and decision variables in each stage are generated when needed, so that the computer memory requirement is significantly reduced.

Description of the Recursive Search Dynamic Programming Method

For a N lines problem with a starting port and ending port, we can connect the two ports and hence change the problem to a N + 1 lines problem. The problem of selecting the shortest route that covers the N + 1 lines can be stated mathematically as follows:

Minimize:

$$D = \sum_{i=1}^{N+1} \sum_{j=1}^{2} \sum_{m=1}^{N+1} \sum_{n=1}^{2} d(i,j,m,n) \cdot X(i,j,m,n) \qquad (3\text{-}1)$$

Subject to:

$$\sum_{\substack{m\neq i\\ m=1}}^{N+1} \sum_{n=1}^{2} X(i,1,m,n) + \sum_{\substack{p\neq i\\ p=1}}^{N+1} \sum_{q=1}^{2} X(i,2,p,q) = 1$$

$$\sum_{\substack{m\neq i\\ m=1}}^{N+1} \sum_{n=1}^{2} X(m,n,i,j) + \sum_{\substack{p\neq i\\ p=1}}^{N+1} \sum_{q=1}^{2} X(i,j,p,q) = 1$$

where

$i = 1, 2, 3, \ldots, N+1 \qquad j = 1, 2$

and

$X(i,j,m,n) = 0$ or 1

given

$d(i,j,m,n)$ = distance from line i, end j to line m, end n

$X(i,j,m,n)$ = 1 if line i, end j is linked to line m, end n

= 0 otherwise

For a N line configuration, there are $2^N N!$ possible routes that would have to be considered for exhaustive enumeration. To solve Equation (3-1) by the dynamic programming method, let the K^{th} stages of the usual dynamic programming formulation correspond to the selection of $(K+1)^{th}$ lines to ending port P_2 that traverses K lines. For example, the first stage corresponds to the selection of a second line after covering first line before return to P_2. The decision variables are then the selection of an alternative route at each stage. The state corresponds to the route line which remains to be selected. For example, at first stage, the decision variables are the selection of an alternative route for covering first line before return to P_2. The state variables are the possible selection of a second line covering previous first line before return to P_2.

The following notation will be used in formulating the dynamic programming model of the problem:

$P_1(m,n)$ = the distance from starting port P_1 to line m, end n.

$d(i,j,m,n)$ = the distance from line i, end j to line m, end n.

$P_2(m,n)$ = the distance from line m, end n to ending port P_2.

$g_k(i,j,m,n,l_k)$ = the distance of path segment from line i, end j, to line m, end n, covering k lines and then return to ending port P_2. l_k denotes a unique combination of k lines from N lines.

$f_k^*(i,j,l_k)$ = the shortest distance of path segment from line i, end j, covering k lines and then returning to P_2.

$F_N^*(P_1,P_2)$ = the shortest distance of a complete route (path) from starting port P_1 to ending port P_2, covering N lines.

The dynamic programming principal of optimality is then implemented by utilizing the following recursive relationship for each stage:

Stage 1

$$f_1^*(i,j,l_1) = \underset{\substack{n=1,2\\ \bar{n}=3-n}}{\text{Minimum}} \left[d(i,j,l_1,n) + P_2(l_1,\bar{n})\right]$$

for the following states:

$i = 1, 2, \ldots, N$

$j = 1, 2$

$l_1 = 1, 2, 3, \ldots, N \qquad l_1 \neq i$

Stage 2

Let

$$g_2(i,j,m,n,l_1) = d(i,j,m,n) + f_1^*(m,\bar{n},l_1)$$

where $n = 1, 2 \qquad \bar{n} = 3-n$

then

$$f_2^*(i,j,l_2) = \underset{\substack{m=1,2,\ldots,N \quad m\neq l\\ n=1,2\\ \text{all } l_1}}{\text{Minimum}} \quad g_2(i,j,m,n,l_1)$$

for the following states:

$i = 1, 2, \ldots, N$

$j = 1, 2$

Stage K

Let

$$g_k(i,j,m,n,l_k) = d(i,j,m,n) + f_{k-1}^*(m,\bar{n},l_{k-1})$$

where

$i = 1, 2, \ldots, N$

$m = 1, 2, \ldots, N \quad m \neq 1$

$j = 1, 2$

$n = 1, 2$

$\bar{n} = 3-n$

then

$$f^*_k(i,j,l_k) = \underset{\substack{m=1,2,\ldots,N \quad m\neq 1 \\ n=1,2 \\ \text{all } l_{k-1}}}{\text{Minimum}} \; g_k(i,j,m,n,l_{k-1})$$

for the following states:

$i = 1, 2, \ldots, N$

$j = 1, 2$

<u>Stage N</u>

$$F^*_N(P_1,P_2) = \underset{\substack{i=1,2,\ldots,N \\ j=1,2 \\ \bar{j}=3-j}}{\text{Minimum}} \left[P_1(i,j) + f^*_{N-1}(i,\bar{j},l_{N-1})\right]$$

Using the conventional dynamic programming method, it is necessary to determine the optimum value of each decision variable for each feasible input state, before calculations are commenced for the next stage. This requires storing approximately $2 \sum_{i=1}^{N} \prod_{j=0}^{i-1} (N-j)$ values, so that a problem with a modest number of lines can easily exceed the memory capacity of the largest computer.

Now assume a starting solution $X = [X(i,j,m,n)]$, for $i = 1, 2, \ldots, N+1$, $j = 1, 2$, $i \neq n$, such that X satisfies the constraints given in Equation (3-1). Although X defines a feasible solution, it is not necessarily the optimum solution. The recursive search technique provides a method of generating the state variable needs one at a time to improve the initial solution until optimum is reached. For example, in a three line configuration (four lines if plus two ports), at first stage, twelve state variables must be generated and stored for the conventional dynamic programming method, while only four state variables are needed to be generated and stored for the recursive search dynamic programming method. A comparison of storage requirements for the conventional dynamic programming method and the recursive search dynamic programming method for a four line configuration is shown in Table I.

TABLE I

A COMPARISON OF VECTOR STORE FOR A FOUR LINE CONFIGURATION PROBLEM

Method	Conventional	Recursive Search
Stage / Vector Store	D.P.	D.P.
1	48	4
2	24	6
3	8	8
Total	80	18

In general, the difference in computer memory storage requirements for N-line configurations is:

$$2 \sum_{i=1}^{N} \prod_{j=0}^{i-1} (N-j) - \sum_{m=2}^{N} 2^m$$

It is obvious that the reduction of storage requirements by using this technique is a trade off with computational time increasing at about the same rate that the computer memory storage is reduced.

Programmed Algorithm

The recursive search dynamic programming algorithm is programmed in the FORTRAN IV language. The logic flow chart of the program is presented in Figure 2. The programmed algorithm will select the optimal route for a configuration of 8 lines or less and requires a computer memory capacity of approximately 62 K bytes. The time required to obtain the optimal path through a N-line prospect configuration is approximately $(2.35)^N(0.001)$ minutes when executed on the IBM 360/65 computer.

Execution times and storage requirements for selecting the optimal route using conventional dynamic programming method by Willard [66] and using the recursive search dynamic programming method of this research is compared and shown in Figures 3 and 4. In Figure 3, the execution times for both methods will increase sharply after 9-line configuration. In Figure 4, the core-storage requirements for both methods will also increase exponentially after 9-line configuration.

The result on both conventional dynamic programming method and recursive search dynamic programming method have shown that using

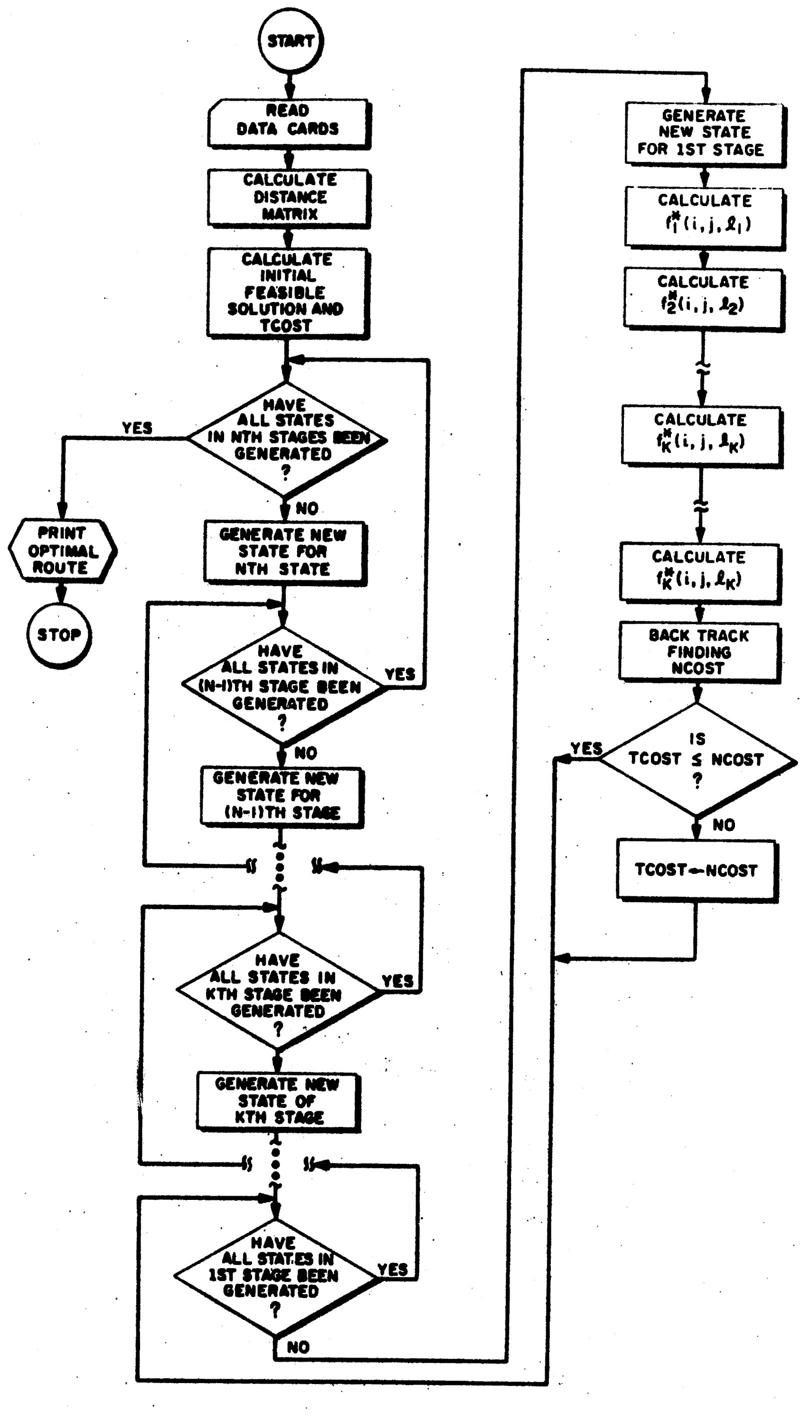

START
READ DATA CARDS
CALCULATE DISTANCE MATRIX
CALCULATE INITIAL FEASIBLE SOLUTION AND TCOST
HAVE ALL STATES IN NTH STAGES BEEN GENERATED ?
YES
NO
PRINT OPTIMAL ROUTE
STOP
GENERATE NEW STATE FOR NTH STATE
HAVE ALL STATES IN (N-1)TH STAGE BEEN GENERATED ?
YES
NO
GENERATE NEW STATE FOR (N-1)TH STAGE
HAVE ALL STATES IN KTH STAGE BEEN GENERATED ?
YES
GENERATE NEW STATE OF KTH STAGE
HAVE ALL STATES IN 1ST STAGE BEEN GENERATED ?
YES
NO
GENERATE NEW STATE FOR 1ST STAGE
CALCULATE $f_1^*(i, j, \ell_1)$
CALCULATE $f_2^*(i, j, \ell_2)$
CALCULATE $f_K^*(i, j, \ell_K)$
CALCULATE $f_K^*(i, j, \ell_K)$
BACK TRACK FINDING NCOST
IS TCOST ≤ NCOST ?
YES
NO
TCOST ← NCOST

Figure 2. The Logic Flow Chart of the Recursive Dynamic Programming Method

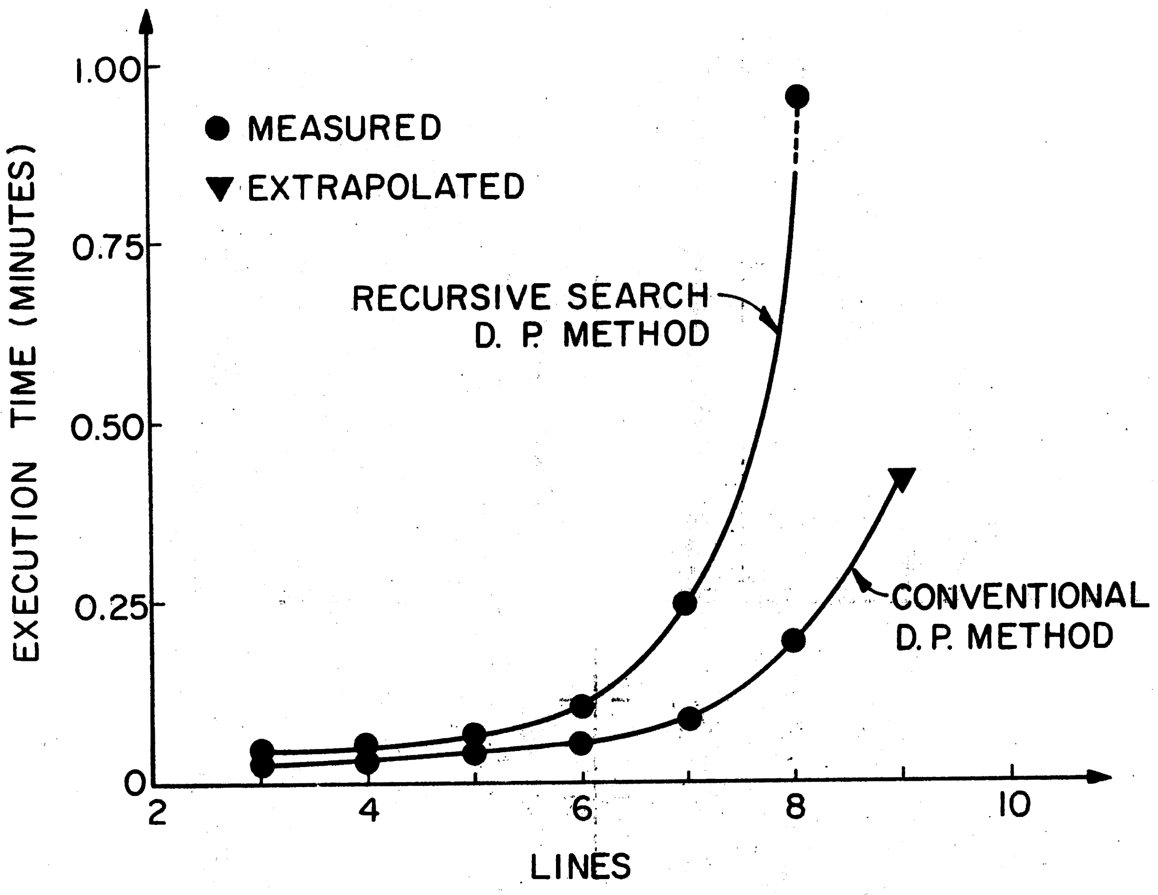


Figure 3. A Comparison of Execution Time of Dynamic Programming Methods

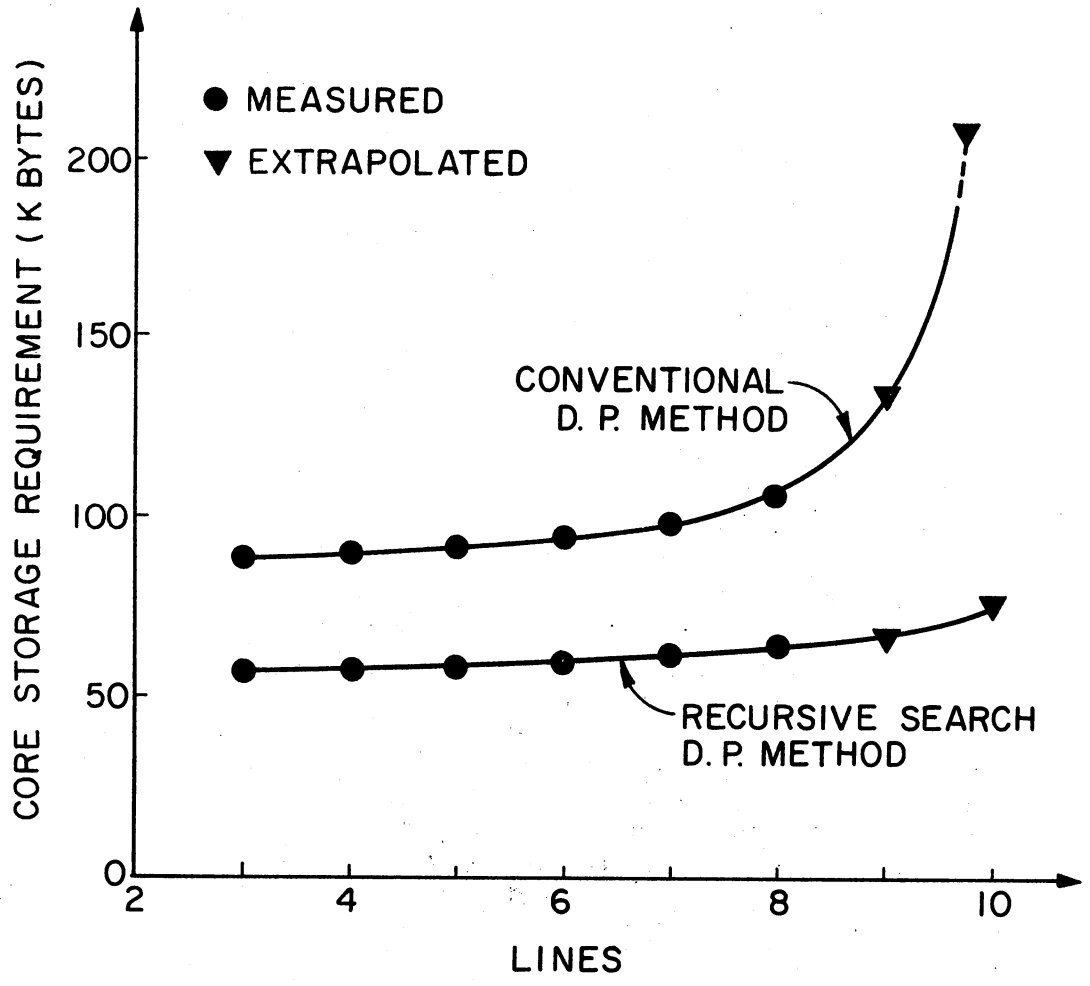


Figure 4. A Comparison of Core Storage Requirement of Dynamic Programming Methods

dynamic programming approach to solve a ship routing problem is limited to a small 9-line configuration on the IBM 360-65 computer. The problem beyond a 9-line configuration, will either exceed computer memory a availability or require a huge amount of computation time. It is concluded from this research that it is not practical to use the dynamic programming approach (either the conventional method or the recursive search method) as a further research tool for a large scale ship routing problem.

CHAPTER IV

ALGORITHMS FOR THE SINGLE-SHIP THROUGH SINGLE-PROSPECT PROBLEM USING A BRANCH AND BOUND APPROACH

As mentioned in the previous literature review, branch and bound, or tree-search techniques, have been used to solve the traveling salesman problem. Among the tree-search algorithms, Little's sequential tour-building method and Eastman's subtour elimination method have shown the greatest promise. Eastman's subtour elimination concept has been extended by Shapiro [59], Bellmore and Malone [8], and Garfinkel [23]. These two methods are studied in this dissertation. The objective is to investigate the applicability of the branch and bound technique to the ship routing problem which can be considered as a constrained traveling salesman problem.

The Constraints of the Problem

The ship routing problem for marine seismic mapping operation is similar to a constrained traveling salesman problem. The constrained traveling salesman problem is defined in this research as follows: Given N cities in which certain cities are grouped together and if one of the cities within a group is visited, then all the other cities in the same group must be visited sequentially find the shortest route that starts at a home city, visits each city once and returns to the home city. Figure 5 shows an example of a constrained traveling salesman problem. One wishes to determine a route which starts from a home city, covers three regions and returns to the home city. In the ship routing problem, the constraint is that each seismic line where the seismic data is actually collected (productive line) has to be traversed as a whole segment. In other words, the two end points of a seismic line must be traversed sequentially. This problem is similar to the constrained traveling salesman problem in which each group contains only two cities. Figure 6 shows an example of a ship routing problem of 3-line configuration.

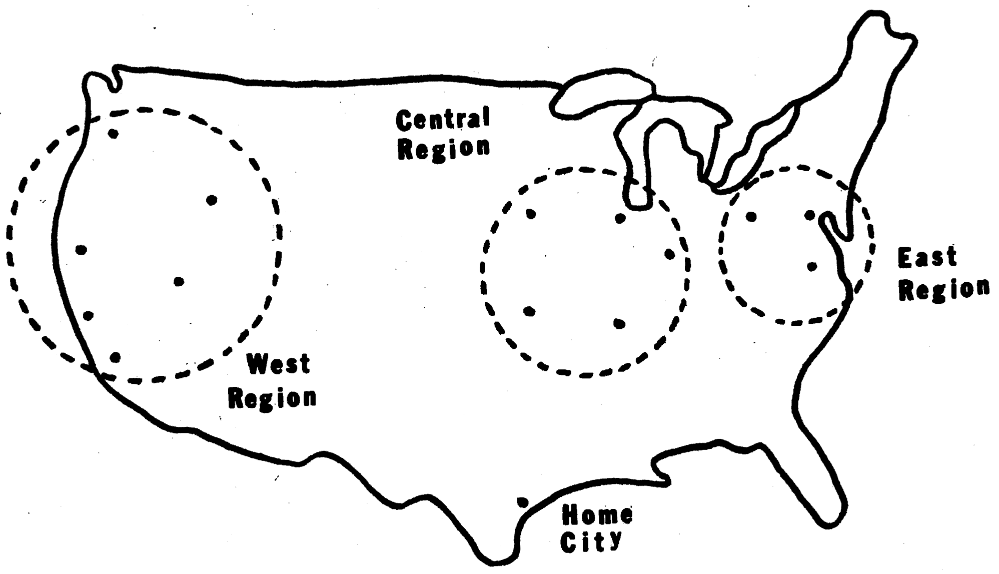


Figure 5. An Example of Constrained Traveling Salesman Problem

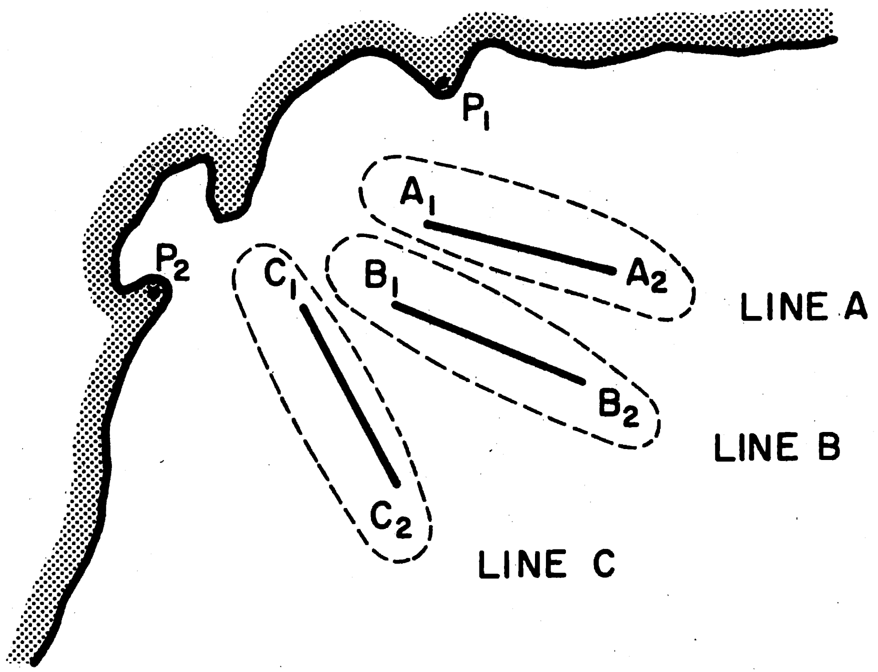


Figure 6. An Example of Ship Routing Problem with 3-Line Configuration

Sequential Tour-Building Algorithm For Ship Routing Problem

Description of the Algorithm

The basis of Little's algorithm is to divide the set of all the possible tours into smaller and smaller subsets and to calculate for each subset a lower bound on the cost of the best tour therein. The object of calculating a bound is two fold:

1) it may be used as a guide for the partitioning of the subsets,

and

2) it limits the search and also identifies the optimal tour (the optimal tour is a tour whose cost is less than or equal to the lower bounds on all the incompletely searched subsets).

In order to apply Little's sequential tour-building concept to the ship routing problem, the following principle is defined:

<u>Principle</u>: Considering two seismic lines in a prospect configuration, one line having end points X_1, X_2 and the other line having end points Y_1, Y_2, if the ship does not go from X_1 to Y_2, then she can go

from X_1 either to X_2 and from there to any point (including Y_2) or to any other point.

to Y_2 either from any other point or from any point (including X_1) to Y_1 and from Y_1 to Y_2.

The algorithm is explained below. It is convenient to represent the partitioning as branching of a tree, where the nodes represent the subsets of tours.

We start with the original cost matrix C, with dimensions $2(N+1)$ by $2(N+1)$. The first two columns and rows are P_1 and P_2. All diagonal elements are set to infinity. Since P_1 is considered the starting port and P_2 the ending port, all the cost elements on the row of P_2 and the column of P_1 are set to infinity. A reducing process is then performed on each row and column. In contrast to Little's original method, the reduction process of this algorithm is accomplished by considering pairs of rows (columns), each pair consisting of the two rows (columns) corresponding to the same line with different end point (say, X_1, X_2, or Y_1, Y_2). The minimum cost in the two rows r_i (columns l_j) is subtracted from all entries in these rows (columns). The resulting matrix C' is then said to be reduced, that is, it contains at least one zero in every row and column pair. The sum of reducing constants is:

$$b_0 = \sum_i^N r_i + \sum_j^N l_j$$

If $Z'(t)$ refers to a tour cost for the reduced matrix C', and $Z(t)$ to a tour cost for the original matrix, and K to the sum of all costs of productive lines where the seismic data is actually collected, then $Z'(t) + K = Z(t) + K - b_0$. Since K is a constant, we can eliminate it from all calculations by setting all productive line costs equal to infinity in the original cost matrix (i.e., $X_1X_2 = \infty$, $X_2X_1 = \infty$, $Y_1Y_2 = \infty$, $Y_2Y_1 = \infty$, ..., etc.) and hence $Z'(t) = Z(t) - b_0$. Since all the elements in the reduced matrix are non-negative, it is clear that $Z'(t) \geq 0$, and hence $Z(t) \geq b_0$, i.e., b_0 is a lower bound on the cost of any tour of the original matrix.

One can now form a tour by selecting one of the links in the reduced matrix which has a zero cost. Rather than selecting one of these links at random, one selects the link whose cost under the reduced matrix is zero and whose penalty is the largest. This penalty cost θ_{X_1,Y_2} (the cost incurred by not going from X_1 to Y_2) for this algorithm is determined as follows:

$$\theta_{X_1,Y_2} = \delta_{X_1} + \delta_{Y_2}$$

where

δ_{X_1} = minimum entry in rows X_1 and X_2, not including X_1Y_2, and

δ_{Y_2} = minimum entry in column Y_1 and Y_2, not including X_1Y_2.

Suppose link (X_1,Y_2) is chosen in this way. The total number of tours is now divided into two subsets, those that include link (X_1,Y_2) and those that do not. These subsets are represented diagrammatically as nodes (X_1Y_2) and $(\overline{X_1,Y_2})$ in Figure 7.

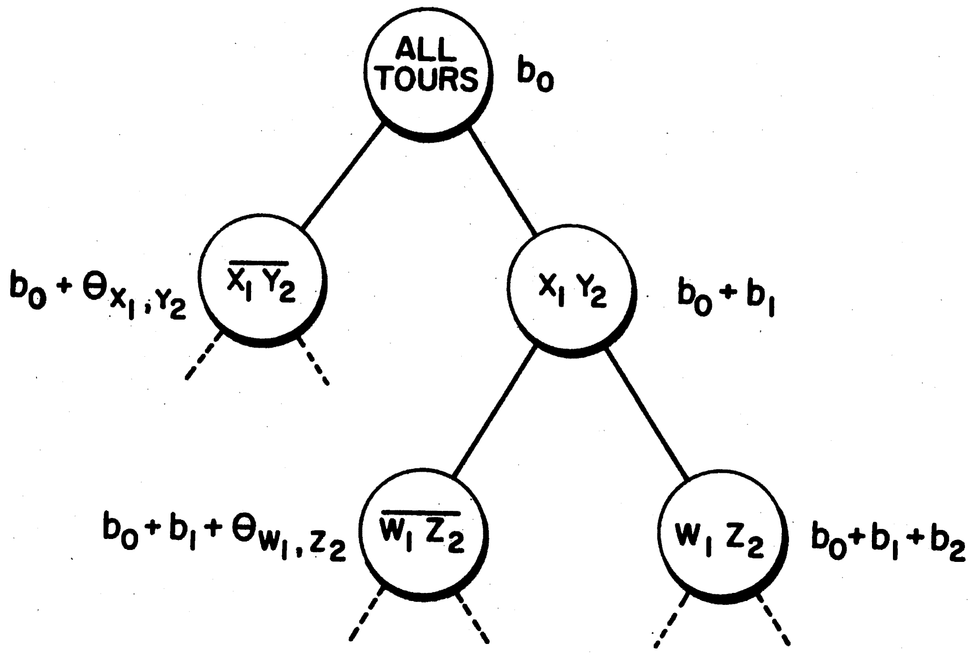


Figure 7. Decision Tree for Branch and Bound Method

The bound for all tours represented at a node is shown marked at the node. Thus since θ_{X_1,Y_2} is the minimum penalty that has to be paid for not including link (X_1,Y_2), the bound on node $(\overline{X_1,Y_2})$ is $b_0+\theta_{X_1,Y_2}$.

The inclusion of link (X_1,Y_2) implies a partial tour consisting of $X_2 \rightarrow X_1 \rightarrow Y_2 \rightarrow Y_1$. Therefore, no other link can emanate from X_2, X_1, Y_2, or finish at X_1, Y_2, Y_1. Thus the row of X_2, X_1, Y_2, and the column of X_1, Y_2, Y_1 can be deleted from the reduced matrix as they will no longer be needed. Also, since the link (Y_1,X_2) is no longer possible as it would create a small subtour, the cost element in the cell (Y_1,X_2) is set to infinity thereby preventing it from being subsequently selected.

The deleting of the row of X_2, X_1, Y_2 and the column of X_1, Y_2, Y_1 produces a matrix with dimensions $[2(N+1)-3]$ by $[2(N+1)-3)$. This matrix is subject to reduction in the same way as the original matrix with the sum of the reducing constants denoted as b_1. If the matrix cannot be reduced, that is, if there exists at least one zero at each row and column pair, then $b_1=0$. At this point one obtains a lower bound of all tours containing link (X_1,Y_2) as b_0+b_1.

A new branching is now in order. Again, new penalties of all the zero elements of the new matrix are calculated for each row and column pair, and a link, say (W_1,Z_2), is selected as the next node. The lower bound for node $\overline{(W_1,Z_2)}$ is $b_0+b_1+\theta_{W_1,Z_2}$ and the lower bound for node (W_1,Z_2) is $b_0+b_1+b_2$, where b_2 is the sum of the reducing constants of the matrix obtained after deleting the row of W_2, W_1, Z_2 and the column of W_1, Z_2, Z_1 (in addition to the row of X_2, X_1, Y_2 and the column of X_1, Y_2, Y_1 which have already been deleted). It should be noted that in some cases the rows and columns subject to deletion at one stage of a branch might overlap with a previously deleted row and column of a prior stage. This is in contrast to Little's original method where if a link is included, the matrix is always reduced by one row and one column. Infinities are again inserted in the appropriate place in the matrix to prevent subloops from ever being formed. In this particular case, since the link (W_1,Z_2) does not join with the previous link (X_1,Y_2), the cost element of the link (W_1,Z_2) is set to infinity. However, if two links form a chain, say the link (Y_1,Z_2) and the link (X_1,Y_2), then the cost element of the link (Z_1,X_2) is set to infinity.

The branching can be continued until the selected links (X_1,Y_2), (W_1,Z_2), ..., etc., form a tour of cost say Z_0. If the lower bounds on all the nodes where branching is possible (i.e., the "free" nodes from which branching has not occurred) are greater than or equal to Z_0, then this is the optimal tour. If not, then any one of the nodes with a bound less than Z_0 can be chosen for further branching.

The choice of which node from which to branch will affect both computing times and computer storage requirements to a very great extent. Two alternative strategies for choosing the node can be used:

1) Branch from the "free" node which has the least bound.

2) Branch from the "free" node nearest to the present node, proceeding upwards in the tree.

In this dissertation, three separate computer programs for the above two strategies are developed to compare the difference of computation times and storage requirements. The detail of these computer programs will be described in the next section of this chapter.

The sequential tour-building algorithm for ship routing problem using the modification of the Little's original method is illustrated by the following simple example.

Example 4-1: Consider the 3-line configuration ship routing problem, where the cost matrix C is given in Figure 8:

	P_1	P_2	A_1	A_2	B_1	B_2	C_1	C_2
P_1	-	41	11	40	43	16	11	34
P_2	41	-	30	11	16	35	35	16
A_1	11	30	-	29	32	11	7	23
A_2	40	11	29	-	5	30	32	8
B_1	43	16	32	5	-	32	34	9
B_2	16	35	11	30	32	-	5	23
C_1	11	35	7	32	34	5	-	24
C_2	34	16	23	8	9	23	24	-

Figure 8. Original Cost Matrix of Example 4-1

<u>Original Cost Matrix Operation</u> (See Figure 9)

1) Calculate the sum of all productive line K.

   $K = (A_1,A_2) + (B_1,B_2) + (C_1,C_2) = 85$

2) Set all productive line equal to infinity.

3) Set all the elements in P_2 row and P_1 column to infinity.

	P_1	P_2	A_1	A_2	B_1	B_2	C_1	C_2
P_1	-	-	11	40	43	16	11	34
P_2	-	-	-	-	-	-	-	-
A_1	-	30	-	-	32	11	7	23
A_2	-	11	-	-	5	30	32	8
B_1	-	16	32	5	-	-	34	9
B_2	-	35	11	30	-	-	5	23
C_1	-	35	7	32	34	5	-	-
C_2	-	16	23	8	9	23	-	-

Figure 9. Reduced Cost Matrix-1

This matrix is not "reduced".

Sub-problem 1 Operation (See Figure 10)

1) Reduce the matrix. Total cost reduced = 32. The bound for sub-problem 1 = 32.

2) Calculate θ_{k_1,l_j} (shown at top-right for each zero cell).

3) Select $\underset{\substack{\text{all } i \\ \text{all } j}}{\text{Max}}\ \theta_{k_i,l_j} = (A_2,P_2)$ as the branch node.

4) For sub-problem 2 (A_2,P_2), delete row: A_1,A_2,P_2 and column: A_2,P_2,P_1, set $P_1A_1 = \infty$.

5) For sub-problem 3 $\overline{(A_2,P_2)}$, set $A_2P_2 = \infty$.

The bound for sub-problem 3 = 32 + 5 = 37.

	P_1	P_2	A_1	A_2	B_1	B_2	C_1	C_2	
P_1	-	-	0^{0}	29	32	5	0^{0}	23	11
P_2	-	-	-	-	-	-	-	-	
A_1	-	19	-	-	27	6	2	18	5
A_2	-	$0^{\boxed{5}}$	-	-	0^{0}	25	27	3	
B_1	-	5	27	0^{0}	-	-	29	4	5
B_2	-	24	6	25	-	-	0^{0}	18	
C_1	-	24	2	27	29	0^{2}	-	-	5
C_2	-	5	18	3	4	18	-	-	
	6		0		0		0		Tot = 32

Figure 10. Reduced Cost Matrix-2

(A_2, P_2)

<u>Sub-problem 2 Operation</u> (See Figures 11 and 12)

1) Impose all necessary constraints.

2) Reduce matrix. Total cost reduced = 2. The bound for sub-problem 2 = 32 + 2 = 34.

3) Calculate $\underset{\substack{\text{all } i \\ \text{all } j}}{\text{Max}}\ \theta_{k_i, l_j} = (C_1, A_1)$ as the next branch node.

4) For sub-problem 4 (P_1, C_1), delete row P_1, C_1 and column C_1, C_2 set $C_2P_2 = \infty$.

5) For sub-problem 5 $(\overline{P_1, C_1})$, set $P_1C_1 = \infty$. The bound for sub-problem 5 = 34 + 5 = 39.

sub prob 2 : $(A_2 \; P_2)$

	A_1	B_1	B_2	C_1	C_2
P_1	-	32	5	0	23
B_1	27	-	-	29	4
B_2	6	-	-	0	18
C_1	2	29	0	-	-
C_2	18	4	18	-	-

Figure 11. Reduced Cost Matrix-3

	A_1	B_1	B_2	C_1	C_2	
P_1	-	32	5	0^{5}	23	0
B_1	25	-	-	29	4	0
B_2	4	-	-	0^{4}	18	
C_1	0^{4}	29	0^{4}	-	-	0
C_2	16	4	18	-	-	
	2		0		0	

Figure 12. Reduced Cost Matrix-4

Sub-problem 4 Operation (See Figures 13 and 14)

1) Impose all necessary constraints.
2) Reduce matrix. Total cost reduced = 9. The bound for sub-problem 4 = 34 + 8 = 42.
3) Select a sub-problem with least bound which is the sub-problem 3.

	A_1	B_1	B_2
B_1	25	-	-
B_2	4	-	-
C_2	16	4	8

Figure 13. Reduced Cost Matrix-5

	A_1	B_1	B_2	
B_1	21	-	-	4
B_2	0	-	-	4
C_2	12	0	4	
	0		0	

Figure 14. Reduced Cost Matrix-6

<u>Sub-problem 3 Operation</u> (See Figure 15)

1) Impose all necessary constraints.

2) Calculate θ_{k_i, l_j}

Select $\underset{\substack{\text{all } i \\ \text{all } j}}{\text{Max}}\ \theta_{k_i, l_j} = (A_2, B_1)$ as the next branch node.

3) For sub-problem 6 (A_2, B_1), delete row: A_1, A_2, B_1 and column: A_2, B_1, B_2, set $B_2A_1 = \infty$.

4) For sub-problem 7 $(\overline{A_2, B_1})$, set $A_2B_1 = \infty$.

The bound for sub-problem 7 = 37 + 2 = 39.

	P_1	P_2	A_1	A_2	B_1	B_2	C_1	C_2
P_1	-	-	0^{0}	29	32	5	0^{0}	23
P_2	-	-	-	-	-	-	-	-
A_1	-	14	-	-	27	6	2	18
A_2	-	-	-	-	0^{2}	25	27	3
B_1	-	0^{0}	27	0^{0}	-	-	29	4
B_2	-	19	6	25	-	-	0^{0}	18
C_1	-	19	2	27	29	0^{0}	-	-
C_2	-	0^{0}	18	3	4	18	-	-

Figure 15. Reduced Cost Matrix-7

Sub-problem 6 Operation (See Figure 15)

1) Impose all necessary constraints.

2) Calculate θ_{k_i,l_j}. Select $\underset{\substack{\text{all } i\\ \text{all } j}}{\text{Max}}\ \theta_{k_i,l_j} = (C_2,P_2)$ as the next branch node.

3) For sub-problem 8 (C_2,P_2), delete row: C_1, C_2, P_2 and column: C_2, P_2, P_1 set $P_1C_1 = \infty$.

4) For sub-problem 9 $\overline{(C_2,P_2)}$, set $C_2P_2 = \infty$.
The bound for sub-problem 9 = 37 + 19 = 56.

	P_1	P_2	A_1	C_1	C_2
P_1	-	-	0^{0}	0^{0}	23
P_2	-	-	-	-	-
B_2	-	21	-	0^{18}	18
C_1	-	19	0^{0}	-	-
C_2	-	0^{19}	16	-	-

Figure 16. Reduced Cost Matrix-8

Sub-problem 8 Operation (See Figure 17)

1) Impose all necessary constraints.

2) Reduce matrix. Total cost reduced = 0.
The bound for sub-problem 8 = 37.

3) Choose P_1A_1 and B_2C_1 to form a tour.

	A_1	C_1
P_1	0	-
B_2	-	0

Figure 17. Reduced Cost Matrix-9

4) Since the bounds of all other sub-problems are greater than the cost of tour in the sub-problem 8, we reach the optimal solution of this example.

The decision tree of this example is shown in Figure 18. The logic flow chart for this algorithm using the modification of Little's branch and bound method is shown in Figure 19 using branching strategy (2). The function of the various boxes in this figure are explained further below:

1) Read in the coordinates for all lines and ports.

2) Calculate distances for productive lines and for non-productive lines.

3) Calculate the sum of all productive lines K and then set the cost entries in matrix C for all productive lines equal to infinity. Delete P_2 row (ending port) and P_1 column (starting port).

4) Set Z_0 (the cost of the best tour so far) to infinity, and number the node as 1 for all tours.

5) Reduce matrix C. Set b_0 = sum of reducing constants. Set W(X) (the lower bound on node X) = b_0.

6) Choose link (K_i, L_j) (line K, end i to line L, end j, where $K = 1,2,\ldots,N$; $L = 1,2,\ldots,N$; $i = 1,2$; $J = 1,2$; $K \neq L$) -- called node Y -- to branch to, so that θ_{K_i,L_j}, is the largest of all the penalties.

7) Branch to the exclusive node $\overline{Y}$ and set its bound $W(\overline{Y}) = W(X) + \theta_{K_i,L_j}$.

8) Branch to Y, delete row $K_{i'}$, K_i, L_j and column K_i, L_j, $L_{j'}$, where $i' = 3-i$ and $j' = 3-j$. Place infinities in the matrix C to prevent subloops from being formed. Reduce the matrix C

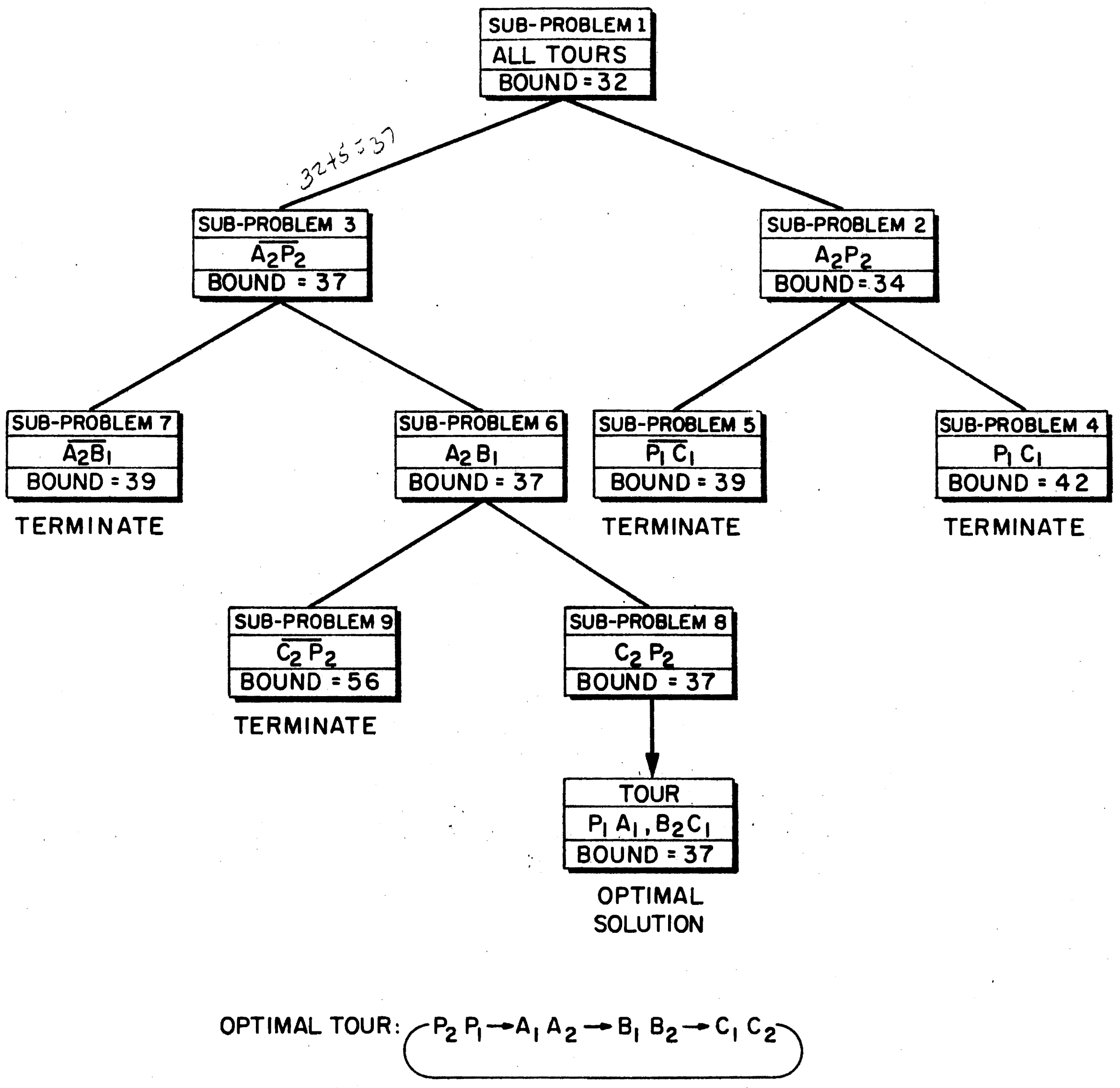


Figure 18. Decision Tree of Example 1

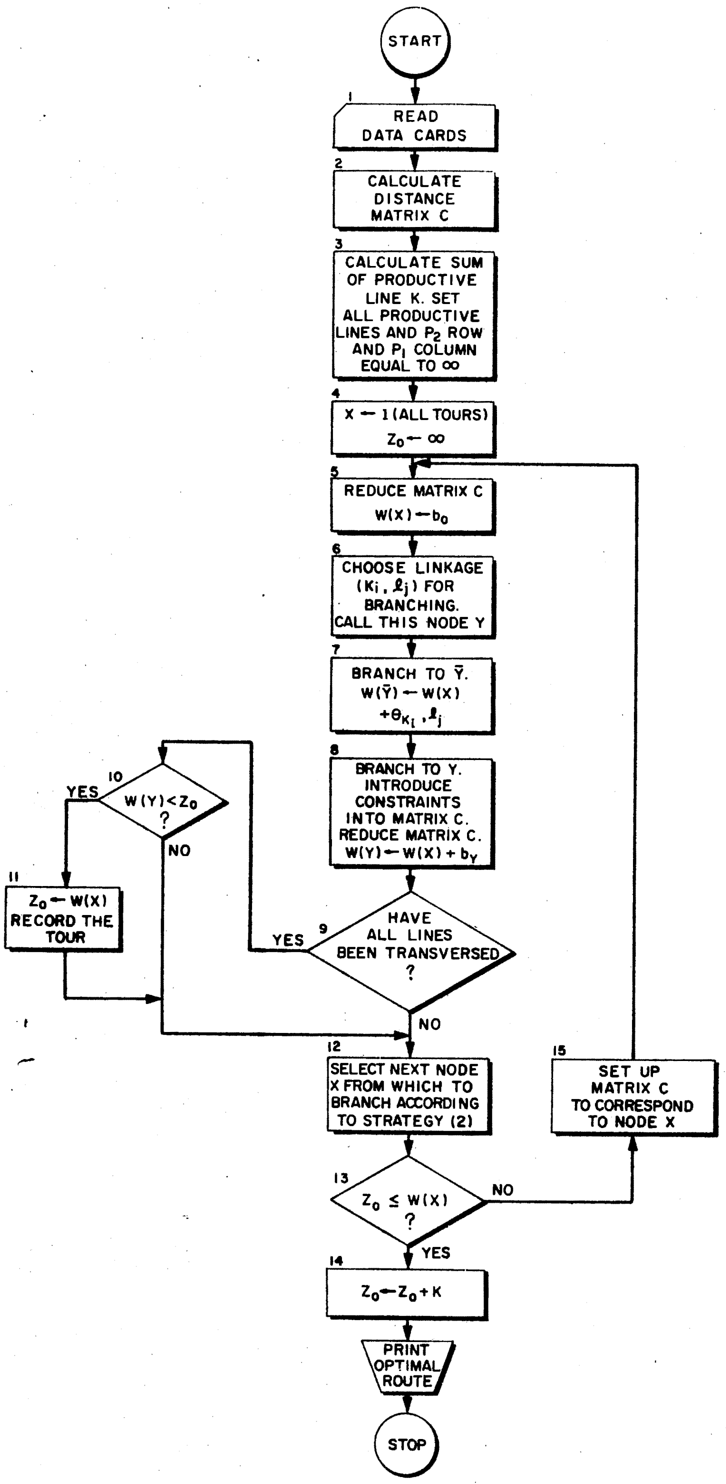

START
1 READ DATA CARDS
2 CALCULATE DISTANCE MATRIX C
3 CALCULATE SUM OF PRODUCTIVE LINE K. SET ALL PRODUCTIVE LINES AND P_2 ROW AND P_1 COLUMN EQUAL TO ∞
4 $X \leftarrow 1$ (ALL TOURS) $Z_0 \leftarrow \infty$
5 REDUCE MATRIX C $W(X) \leftarrow b_0$
6 CHOOSE LINKAGE (K_i, ℓ_j) FOR BRANCHING. CALL THIS NODE Y
7 BRANCH TO $\bar{Y}$. $W(\bar{Y}) \leftarrow W(X) + \theta_{K_i, \ell_j}$
8 BRANCH TO Y. INTRODUCE CONSTRAINTS INTO MATRIX C. REDUCE MATRIX C. $W(Y) \leftarrow W(X) + b_Y$
9 HAVE ALL LINES BEEN TRANSVERSED ?
YES
NO
10 $W(Y) < Z_0$?
YES
NO
11 $Z_0 \leftarrow W(X)$ RECORD THE TOUR
12 SELECT NEXT NODE X FROM WHICH TO BRANCH ACCORDING TO STRATEGY (2)
13 $Z_0 \leq W(X)$?
NO
YES
14 $Z_0 \leftarrow Z_0 + K$
15 SET UP MATRIX C TO CORRESPOND TO NODE X
PRINT OPTIMAL ROUTE
STOP

Figure 19. The Logic Flow Chart of the Algorithm for Ship Routing Problem Using the Modification of Little's Branch and Bound Method

and find b_Y, the sum of the reducing constant. Set the bound on node Y as $W(Y) = W(X) + b_Y$.

9) If all lines have been traversed, a tour has been obtained, go to (10). Otherwise, go to (12).

10) If the cost of this tour W(Y) is less than Z_0, go to (11). Otherwise, go to (12).

11) Record the new tour and update Z_0. Go to (12).

12) Select the next node X from which to branch, as that node with the lowest bound W(X).

13) If all the bounds are greater than Z_0, the branching is completed, the stored tour is optimal, go to (14); otherwise, continue to (15).

14) Add the K value to Z_0 and stop.

15) The matrix C must be updated and set up to correspond to node X as follows. Copy the original cost matrix into the matrix C. Find $T = \Sigma(K_i, L_j)$, going from the top of tree to node X. For each such link (K_i, L_j), delete row K_i', K_i, L_j and column K_i, L_j, L_j' of the matrix C, and set infinities into the appropriate cells in the matrix C to prevent subloops. For each exclusive node in the chain from the top of the tree to node X, set the corresponding element in the matrix C to infinity. Reduce the matrix C and set a new bound on X as $W(X) = T +$ the sum of the reducing constants. Go to step (6).

Programmed Algorithm

The algorithm for the single-ship through single-prospect routing problem using the modification of Little's sequential tour-building

2 stategies, see p. 30

programmed in the FORTRAN IV language. Two separated programs were coded for strategy (2) of this algorithm, which branches a sub-problem from the "free" node nearest to the present node, proceeding upwards in the tree. The first version program of the strategy (2) uses N of N by N arrays to store the information needed at each branching point so that the branches that have been searched can be thrown away. The second version program of the strategy (2) uses only one N by N array and two other one-dimensional arrays to store the information needed. The information is packed when it is stored and unpacked when it is retrieved, in these two one-dimensional arrays. In addition, one program was coded for strategy (1) which branches a sub-problem from the "free" node with least bound. The procedure to store the information needed in this program is the same as the second version program on the strategy (2).

A comparison of execution times and storage requirements on the IBM 360/65 computer for these three programmed algorithms is shown in Figure 20 and Table II. The second version program of strategy (2) is utilized in this algorithm since it takes shorter computation time than the strategy (1) and requires less core storage than the first version of strategy (2).

Subtour Elimination Algorithm for Ship Routing Problem

The idea of Eastman's subtour elimination method is based on the fact that every solution to the traveling salesman problem is also a feasible solution to the corresponding assignment problem. The optimal solution to the assignment problem is therefore a lower bound on the

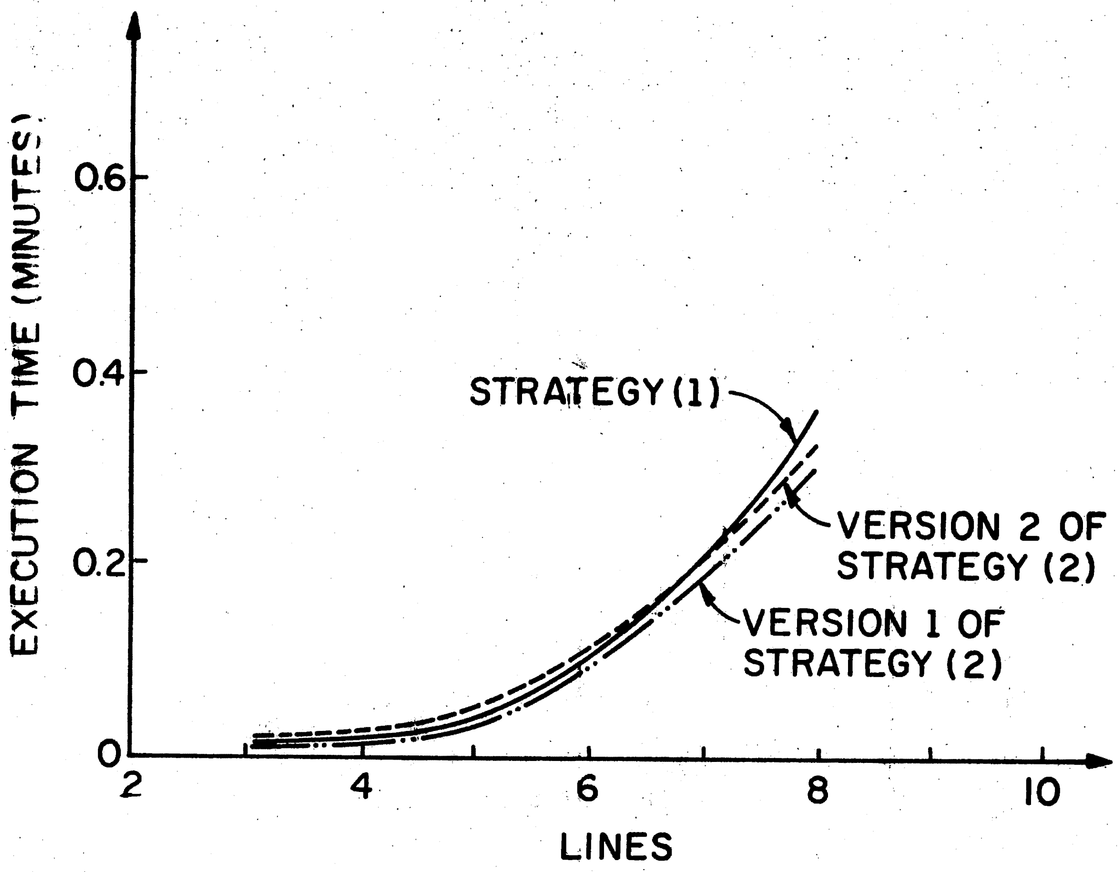


Figure 20. A Comparison of Computation Times of Three Programmed Algorithms

TABLE II

A COMPARISON OF CORE STORAGE REQUIREMENTS AND COMPUTATION TIMES OF THREE PROGRAMMED ALGORITHMS

Time and Storage / Method	Approximate Computation Time (min.)	Core Storage (K Bytes)	Maximum of Lines
Version 1 Strategy (2)	$(2.05)^N \cdot (0.001)$	92	14
Version 2 Strategy (2)	$(2.06)^N \cdot (0.001)$	62	14
Stragegy (1)	$(2.08)^N \cdot (0.001)$	62	14

solution of the traveling salesman problem; and, if the solution is cyclic, it is also the optimal solution to the traveling salesman problem. If the solution is not cyclic, then one or more subtours must exist. Special methods are used to eliminate the subtours by joining them to form one tour that passes through all the N points.

If one applies Eastman's subtour elimination method to the ship routing problem difficulties might arise from the fact that the assignment problem procedure assumes the independence of each entry in cost matrix. The modification of the assignment problem procedure to include the dependence of entries in a cost matrix is beyond the scope of this research. The author's research work in this approach is concentrated on the development of a procedure which, when incorporated with the assignment problem procedure, can be used to solve the ship routing problem.

One attempt of this approach is to set the entries for each line pair (i.e., A_1A_2, A_2A_1, B_1B_2, B_2B_1, ..., etc.) in the original cost matrix equal to infinity. It is hoped that with this kind of set-up, the final optimal solution will include at least one entry out of each line pair so that the feasibility of the solution is maintained. The cost matrix of Example 1 in the previous section with this kind of set-up is in Figure 21.

	P_1	P_2	A_1	A_2	B_1	B_2	C_1	C_2
P_1	-	0	11	40	43	16	11	34
P_2	0	-	30	11	16	35	35	16
A_1	11	30	-	0	32	11	7	23
A_2	40	11	0	-	5	30	32	8
B_1	43	16	32	5	-	0	34	9
B_2	16	35	11	30	0	-	5	23
C_1	11	35	7	32	34	5	-	0
C_2	34	16	23	8	9	23	0	-

entries are zero, not infinity as stated on prev. page.

Figure 21. The Cost Matrix of Example 1 with Special Set-Up for Line Pair

Unfortunately, it can be shown that both entries of a line pair can be forced away, and hence none of the entries on a line pair will be included in the final optimal solution. This situation will occur when some zeroes are generated in a later iteration in the same column or row where the line pair is located. A more effective procedure to explicitly impose the sensitibility constraints of the problem are discussed in the following section.

Description of the Algorithm

This algorithm starts with an original cost matrix C which has

2(N+1) by 2(N+1) elements in it. All diagonal elements of this matrix are set to infinity. The cost C_{K_i,L_j} is interpreted as the cost of traveling from line K, end i to line L, end j. The cost matrix C is then restructured as C', containing only (N+1) by (N+1) elements, by selecting the minimum cost from the four combinations of each two line pair, i.e.,

$$\min_{\substack{i=1,2\\ j=1,2}} C_{K_i,L_j} \qquad \text{for } K = 1,2,\ldots,N+1 \quad L = 1,2,\ldots,N+1 \quad K \neq L$$

The purpose of restructuring the cost matrix C to the cost matrix C' is to definitely include the productive line in the final solution "implicitly". The algorithm then determines the optimal solution utilizing the cost matrix C'. If the solution to the assignment problem is a tour, then this is the optimal solution of the ship routing problem. If the solution to the assignment problem is not a tour, then it must consist of several

1) subtour lines and/or
2) overlapping linkages.

For these cases special procedures are developed to eliminate the subtour lines and overlapping linkages.

Example 4-1 from the previous section is used to illustrate this procedure. Figure 22 shows the reduced cost matrix C' of this example. The letter on the top-left corner of each cost cell indicates the end points of a seismic line where the minimum cost is chosen. For instance, in AB cell, 21 means the 5 is the cost of traveling from line A, end 2 to line B, end 1, which is the smallest among the four combinations of AB pair.

$$\begin{array}{c|c|c|c|c|}
 & P & A & B & C \\
\hline
P & - & \boxed{11}\ 11 & \boxed{12}\ 16 & \boxed{11}\ 11 \\
\hline
A & \boxed{22}\ 11 & - & \boxed{21}\ 5 & \boxed{11}\ 7 \\
\hline
B & \boxed{12}\ 16 & \boxed{12}\ 5 & - & \boxed{21}\ 5 \\
\hline
C & \boxed{11}\ 11 & \boxed{11}\ 7 & \boxed{11}\ 5 & - \\
\hline
\end{array}$$

Figure 22. The Reduced Cost Matrix C′ of Example 4-1

The solution of assignment problem for the cost matrix C′ is shown below with one subtour line and one overlapping linkage.

$$\left.\begin{array}{l} PA \longrightarrow P_1A_1 \\ AP \longrightarrow A_2P_2 \end{array}\right\} \text{SUBTOUR LINES}$$

$$\left.\begin{array}{l} BC \longrightarrow B_2C_1 \\ CB \longrightarrow C_1B_1 \end{array}\right\} \text{OVERLAPPING LINKAGES}$$

The procedure of eliminating subtour lines and the procedure of eliminating overlapping linkages is discussed in detail in the following two sub-sections.

The Procedure of Eliminating Subtour Lines

To eliminate a subtour, Eastman [20] chose one of the subtours having minimum number of linkages, and created a set of new problems for eliminating linkages from subtour. Bellmore and Malone [8] summarized Eastman's approach into the following statement:

> The elimination of any given subtour S of length K can be accomplished by imposing the constraint $\Sigma_{(i,j)\in S'} X_{ij} \leq K-1$, where S' is the ordered-pair representation of S.

Bellmore and Malone [8] extended Eastman's subtour elimination method to a more effective way. Their extension is summarized in the following statement:

> The elimination of any given subtour S of length K can be accomplished by imposing the constraint $\Sigma_{i\in S} \Sigma_{j\in\overline{S}} X_{ij} \geq 1$, where S is the set of cities on the subtour, and $\overline{S}$ is the complement of S.

Garfinkel [23] develops a branching scheme for partitioning the feasible set which has proved to be better than Bellmore and Malone's method in computation times. Therefore, in this research, Garfinkel's method is used to eliminate subtour linkages. Garfinkel's method, which is based on vertex partition, has been extended in this dissertation to a method which can be used to partition edge. The detail and proof of optimality of this method will be discussed in Chapter V. We will briefly review Garfinkel's method [23] as follows.

For a graph $G^K = (V,E^K)$, the subset of tours at V_K of the enumeration tree is denoted by S_K. To separate S_K, choose any vertex set $\theta \subset V$ (there exist at least two), which corresponds to a subtour. If $\theta = \{i_1, \ldots, i_P\}$ and $\overline{\theta}$ is its complement, we impose the constraint $\Sigma_{i\in\theta} \Sigma_{j\in\overline{\theta}} X_{ij} \geq 1$ by separating S_K into P subsets. Each subset corresponds to a graph having some of the edges of E^K deleted. For an arbitrary vertex $i\in\theta$, define:

$$\theta_i = \{(i,j)\,|\,j\in\theta\} \quad \text{and} \quad \theta_i' = \{(i,j)\,|\,j\in\overline{\theta}\}\ .$$

Then the edge sets corresponding to the separation S_K^* are:

$$E_{i_1}^K = E^K - \theta_{i_1}$$

$$- - - - - - - - - - - - - - - - - - - -$$

$$E_{i_2}^K = E^K - \theta_{i_1}' - \ldots - \theta_{i_{P-1}}' - \theta_{i_P}$$

Garfinkel's theorem [23] then states:

Every tour t that is contained in $G_K^K = (V,E^K)$ is contained in exactly one of the graphs $G_{ij}^K = (V,E_{ij}^K)$, $j = 1, \ldots, P$.

Garfinkel's method can now be used to eliminate the subtour lines. In the previous Example 4-1, suppose the solution of the assignment is:

PA ——	P_1A_1
AP ——	A_2P_2
BC ——	B_2C_1
CB ——	C_2B_1

The subtour linkage are A_2P_2 & P_1A_1 and B_2C_1 & C_2B_1. The general procedure of finding a partition subset in order to eliminate the subtour is stated as follows:

1) Select a subtour line with minimal length (in this case, suppose we select P and A). Form θ_i and θ_i' as described previously in Garfinkel's method. (In this case, $\theta_P = \{(PA)\}$; $\theta_P' = \{(PB),(PC)\}$; $\theta_A = \{(AP)\}$; $\theta_A' = \{(AB),(AC)\}$.)

2) For each θ_i and θ_i', assign to it its end point index. (In this case, assume the end point index for PA is 11, PB is 12, PC is 11, AP is 22, AB is 21, AC is 11, then

$$\theta_P = \{(P_1A_1)\}$$
$$\theta_P' = \{(P_1B_2),(P_1C_1)\}$$
$$\theta_A = \{(A_2P_2)\}$$
$$\theta_A' = \{(A_2B_1),(A_1C_1)\} \text{ .)}$$

3) Form the partition subset as described by Garfinkel. (In this case, $E_P = E - \theta_P$

$$E_A = E - \theta_P' = \theta_A \text{ .)}$$

The decision tree diagram for this example with above procedure is shown in Figure 23.

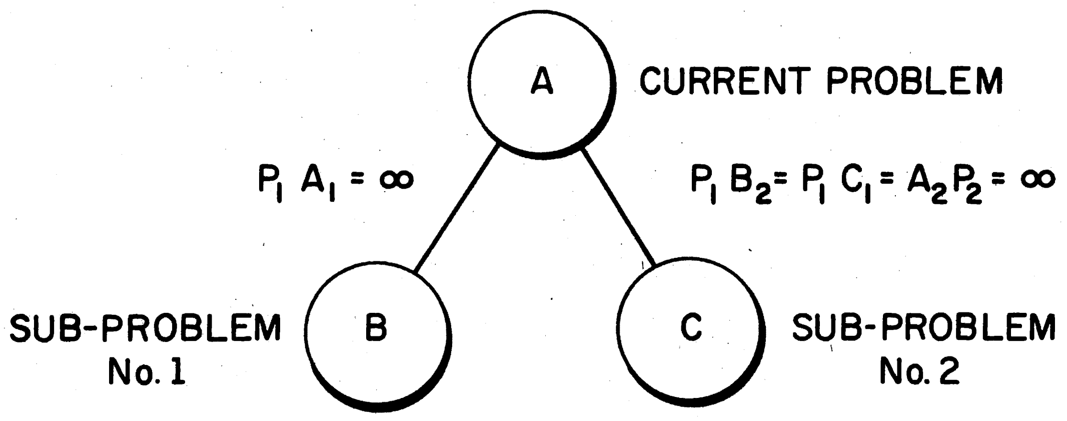


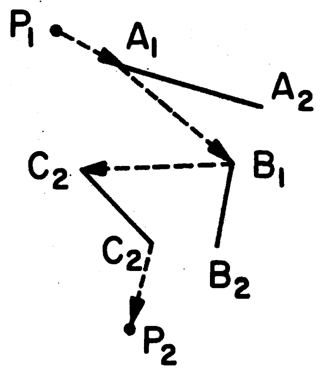
Figure 23. The Decision Tree for a Partition Subset of Subtour Lines

The Procedure of Eliminating Overlapping Linkages

In the exercise of this algorithm, an overlapping linkage will occur at the end point where a linkage is entered and another linkage is exited at same time. For example, if the solution of assignment problem of the previous Example 4-1 is:

$$PA \longrightarrow P_1A_1$$
$$AB \longrightarrow A_1B_1$$
$$BC \longrightarrow B_1C_2$$
$$CP \longrightarrow C_1P_2$$

OR GRAPHICALLY



Overlapping linkages occur at the end point of A_1 and B_1. There are two ways to break the overlapping linkages at the end point of A_1 and B_1, i.e., either A_1B_1 will not be in the final solution or P_1A_1 and B_1C_2 will not be in the final solution. The decision tree diagram for this example is shown in Figure 24.

The general procedure for eliminating an overlapping linkage can be stated as the follows:

1) Select the subtour with minimum length which contains at least one overlapping linkage (there may be only one subtour in the solution of an assignment problem).

2) Find the overlapping linkage in this subtour.

3) Find a mutually exclusive subset for these overlapping linkages that can be used to branch the current problem into more sub-problems.

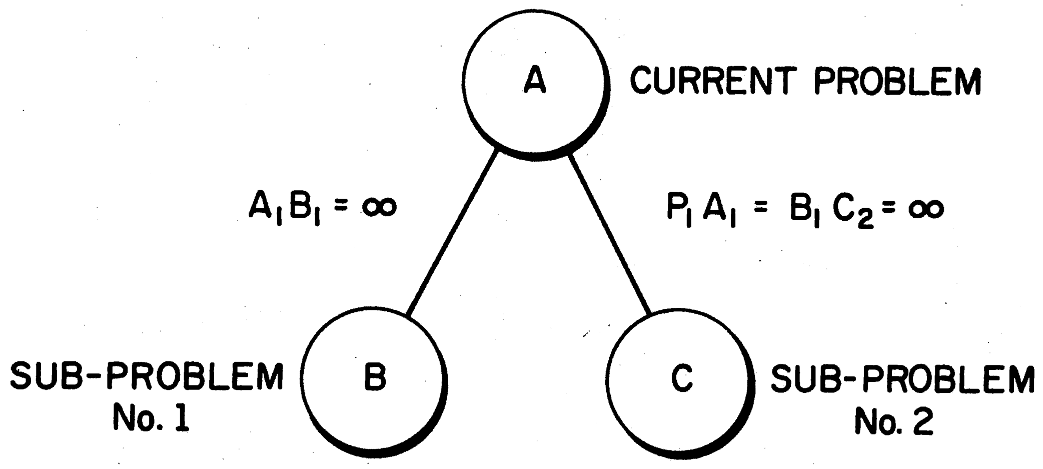


Figure 24. The Decision Tree for Partition Subset of Overlapping Linkages

The algorithm developed to eliminate a subtour line and/or an overlapping linkage for any unsearched sub-problem can now be summarized as the follows:

1) Consider the solution of the current assignment problem. If there exists at least one overlapping linkage in the solution, go to step (2). If no overlapping linkages exist but at least one subtour line is in the solution, go to step (3).
If there exist no overlapping linkages and no subtour lines,

a tour is found. Store the cost and the linkages of this tour and continue to next unsearched sub-problem with least bound.

2) Branch the current problem into more sub-problems by using the procedure of eliminating overlapping linkages, and then continue to next unsearched sub-problem with least bound.

3) Branch the current problem into more sub-problems by using the procedure of eliminating subtour lines, and then continue to next unsearched sub-problem with least bound.

A logic flow chart of this algorithm using the modification of subtour elimination method is shown in Figure 25.

Programmed Algorithm

The algorithm for the single-ship through single-prospect routing problem using the modification of subtour elimination method was programmed in the FORTRAN IV language. The programmed instructions implement the theory presented in the preceding section.

The algorithm requires a computer memory of approximately 82 K bytes for a 14-line configuration. The time required to obtain the optimal path through a N-line prospect configuration is approximately $(2.92)^N(0.001)$ minutes when executed on the IBM 360/65 computer. Since the execution time increases exponentially as the number of lines increases, it is found that this algorithm is not an effective one for solving a large scale ship routing problem. A comparison of this algorithm with other algorithms is discussed in Chapter VII.

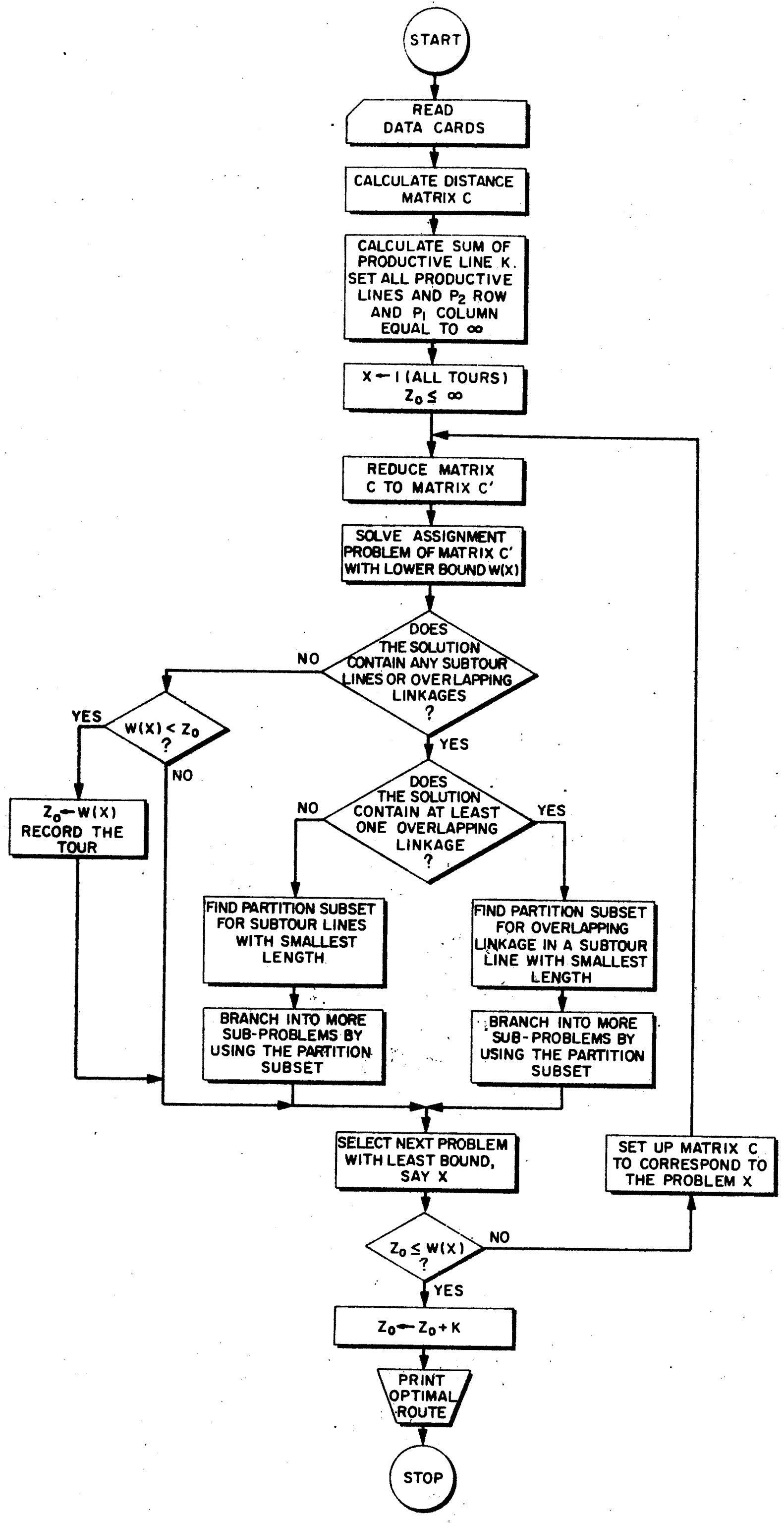

START
READ DATA CARDS
CALCULATE DISTANCE MATRIX C
CALCULATE SUM OF PRODUCTIVE LINE K. SET ALL PRODUCTIVE LINES AND P_2 ROW AND P_1 COLUMN EQUAL TO ∞
X ← 1 (ALL TOURS) $Z_0 \leq \infty$
REDUCE MATRIX C TO MATRIX C′
SOLVE ASSIGNMENT PROBLEM OF MATRIX C′ WITH LOWER BOUND W(X)
DOES THE SOLUTION CONTAIN ANY SUBTOUR LINES OR OVERLAPPING LINKAGES ?
NO
YES
W(X) < Z_0 ?
YES
NO
Z_0 ← W(X) RECORD THE TOUR
DOES THE SOLUTION CONTAIN AT LEAST ONE OVERLAPPING LINKAGE ?
NO
YES
FIND PARTITION SUBSET FOR SUBTOUR LINES WITH SMALLEST LENGTH
FIND PARTITION SUBSET FOR OVERLAPPING LINKAGE IN A SUBTOUR LINE WITH SMALLEST LENGTH
BRANCH INTO MORE SUB-PROBLEMS BY USING THE PARTITION SUBSET
BRANCH INTO MORE SUB-PROBLEMS BY USING THE PARTITION SUBSET
SELECT NEXT PROBLEM WITH LEAST BOUND, SAY X
SET UP MATRIX C TO CORRESPOND TO THE PROBLEM X
$Z_0 \leq$ W(X) ?
NO
YES
Z_0 ← Z_0 + K
PRINT OPTIMAL ROUTE
STOP

Figure 25. The Logic Flow Chart of the Algorithm for Ship Routing Problem Using the Modification of Subtour Elimination Method

CHAPTER V

ALGORITHM FOR THE SINGLE-SHIP THROUGH SINGLE-PROSPECT PROBLEM USING A GRAPH THEORETIC APPROACH

The ship routing problem of finding the shortest route starting from port P_1, traversing all lines and returning to port P_2 is similar to finding the shortest Hamiltonian chain (SCH) of a link-weighted graph. A Hamiltonian chain (HC) is defined as a line passing through every vertex of the graph exactly once. If the link joining the two end vertices of a HC is added, the resulting tour is called a Hamiltonian circuit.

It had been observed by Obruca [47] that the majority of lines appearing in a shortest spanning tree for any network also appear in an optimal solution to the corresponding traveling salesman problem. Christofides [10] used the same idea to develop an algorithm that finds a shortest Hamiltonian chain of a graph. In this dissertation, it is proved that the Christofides' algorithm is ineffective because the partition of a feasible subset is not mutually exclusive. A better procedure has been developed and proved to be optimal. This procedure is then modified and applied to the ship routing of marine seismic mapping operations.

Construction of Shortest Spanning Tree

There are two well known methods that can be used to construct the shortest spanning tree of a graph: These are Prim's method [51] and Kruskal's method [38]. Since the Kruskal's method fits the requirement for constructing a "conditional shortest spanning tree" (which will be discussed later), Kruskal's method and the Seppanen's [58] programmed algorithm are summarized below.

Kruskal's method [38] starts with an original cost matrix C, and first sorts all edges into ascending numerical order. If the cost matrix C is symmetric, then only upper right triangle of the matrix need to be considered. The procedure generates a tree by including the shortest edge which does not form loops with the edges already in the tree. This step is performed as many times as necessary until all edges in the list are exhausted. The spanning tree generated by this procedure has a minimum cost.

Seppanen [58] gives an efficient computer program for Kruskal's algorithm. The main concept of this algorithm is described briefly as follows: At each stage of the algorithm, one edge at a time is considered from rest list of edges, whereby one of four possible conditions will arise. If neither of the vertices is included in a tree, this edge is taken as new tree and its vertices numbered by an incremented component number. If one vertex is in a tree, the edge will be added to this tree. If the two vertices are in two different trees, these will be combined into a single tree by renumbering the vertices of the other component. Finally, if both vertices are in the same tree, the edge completes a fundamental cycle of the graph with respect to the spanning tree and consequently will not be considered further. At the end, the indices of edges in the spanning tree are stored in an array. The procedure will generate a shortest spanning tree of a graph.

The Seppanen's shortest spanning tree computer program, originally coded in the ALGOL language, is re-coded into the FORTRAN IV language with modifications for solving the ship routing problem.

A Review and Analysis of Christofides' Method of Finding a Shortest Hamiltonian Chain

Christofides' method [10] starts with the original cost matrix C in which all diagonal elements are set to infinity. Then the method uses Kruskal's [38] algorithm to find a shortest spanning tree. Within the solution of shortest spanning tree, if the starting and ending vertices have degree of 1 and all other vertices have degree of 2, then the problem is solved. Otherwise, those vertices with degree greater than 2 are candidates for branching. Since at least one of the edges associated with such a vertex must be absent from the final solution, Christofides selects and branches on the vertex with highest degree (> 2). The Christofides' algorithm can be summarized as follows:

Step 1: Find a shortest spanning tree for cost matrix C_{ij}. Obtain cost of shortest spanning tree (C_{SST}) which is the lower bound for the length of shortest Hamiltonian chain. Go to Step 2.

Step 2: If all the vertices have degree of 2 except the starting and ending vertex have degree 1, go to Step 3. Otherwise, select the vertices have degree 1, go to Step 3. Otherwise, with it (say m), and branch m sub-problems from the current problem by excluding each of these m edges one at a time from the initial problem. Go to Step 3.

Step 3: If the bound for an existing shortest Hamiltonian chain is less than the bound for all other unsearched sub-problems, stop. Otherwise, select the sub-problem with least bound and go to Step 1.

An example of this procedure from Christofides' [10] is shown in Figure 26 through Figure 28.

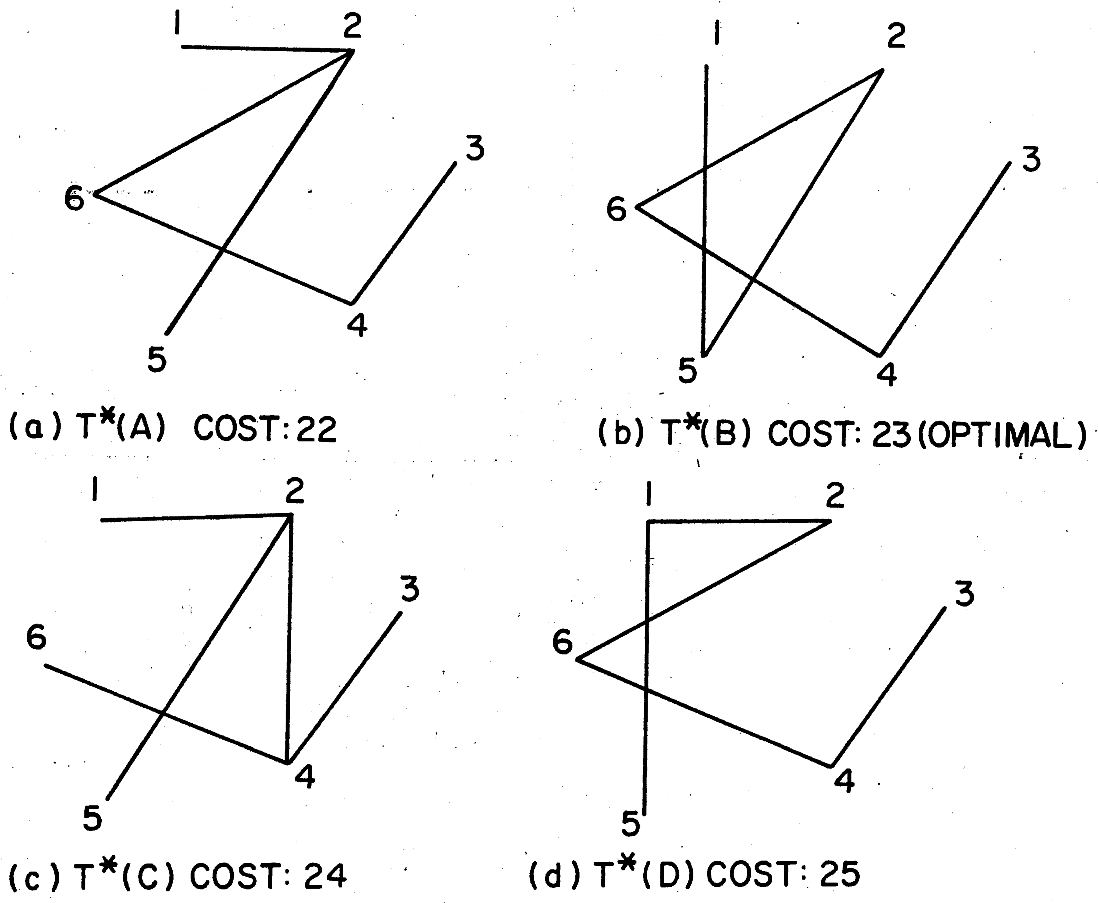


Figure 26. The Shortest Spanning Tree

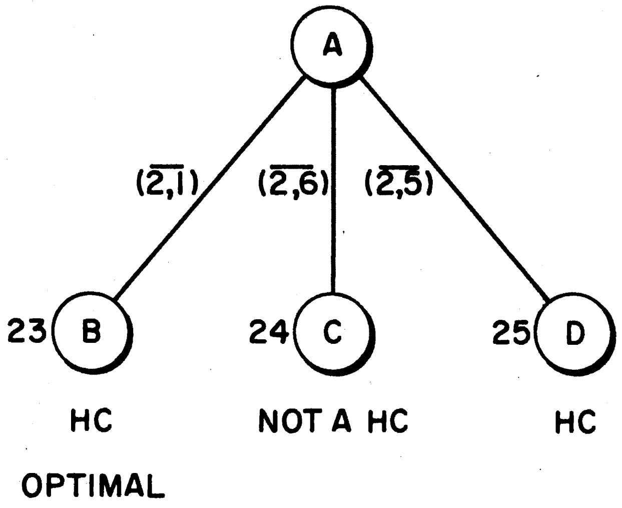


Figure 27. The Decision Tree Search

$$\{c_{ij}\} \equiv \begin{array}{c|cccccc} & 1 & 2 & 3 & 4 & 5 & 6 \\ \hline 1 & 0 & 4 & 10 & 18 & 5 & 10 \\ 2 & 4 & 0 & 12 & 8 & 2 & 6 \\ 3 & 10 & 12 & 0 & 4 & 18 & 16 \\ 4 & 18 & 8 & 4 & 0 & 14 & 6 \\ 5 & 5 & 2 & 18 & 14 & 0 & 16 \\ 6 & 10 & 6 & 16 & 6 & 16 & 0 \end{array}$$

Figure 28. The Cost Matrix

Computational experience with this algorithm has not been reported by Christofides. However, in this dissertation, a computer program based on this algorithm to solve the ship routing problem has been coded and tested. The results on execution time are not encouraging. The weakness of this algorithm is that the branching strategy does not separate the solution spaces into mutually exclusive subsets. The above example will be used to illustrate this point.

According to the spanning tree in Figure 26(c), there are a total of eight possible combinations for generating sub-problems from the three edges associated with vertex 2, namely:

			Combination Index
$(1\text{-}2)_{\text{In}}$	$(2\text{-}6)_{\text{In}}$	$(2\text{-}5)_{\text{In}}$	1
		$(2\text{-}5)_{\text{Out}}$	2
	$(2\text{-}6)_{\text{Out}}$	$(2\text{-}5)_{\text{In}}$	3
		$(2\text{-}5)_{\text{Out}}$	4
$(1\text{-}2)_{\text{Out}}$	$(2\text{-}6)_{\text{In}}$	$(2\text{-}5)_{\text{In}}$	5
		$(2\text{-}5)_{\text{Out}}$	6
	$(2\text{-}6)_{\text{Out}}$	$(2\text{-}5)_{\text{In}}$	7
		$(2\text{-}5)_{\text{Out}}$	8

where $(1\text{-}2)_{\text{In}}$ means edge 1-2 will be included in the new sub-problem;

$(1\text{-}2)_{\text{Out}}$ means edge 1-2 will not be included in the new sub-problem.

Using Christofides' algorithm, three sub-problems are generated and their partitions are:

Problem D (2-5 = ∞): This includes the sub-problems with combination index {2, 4, 6, 8}.

Problem C (2-6 = ∞): This includes the sub-problems with combination index {3, 4, 7, 8}.

Problem B (1-2 = ∞): This includes the sub-problems with combination index {5, 6, 7, 8}.

It is obvious that the partition for each sub-problem is not mutually exclusive and hence, the procedure may not be efficient.

Shortest Hamiltonian Chain Algorithm for the Ship Routing Problem

An Extension of Garfinkel's Method of Partitioning a Feasible Subset

Garfinkel's method for partitioning a feasible subset described in the previous chapter is based upon the discrimination of vertices. In order to apply this method to branch the solution from a shortest spanning tree, we have to extend this procedure from one of discriminating vertices to a procedure of discriminating edges. In other words, instead of partitioning a vertex set, an edge set will be partitioned to generate mutually exclusive sub-problems. A theorem which is similar to Garfinkel's is derived. It is proved that the optimal solution will fall in one of the subsets obtained via the partitioning of the edge set.

For a graph $G^K = (V,E^K)$, form a shortest spanning tree for this graph. Let an edge set of the tree be E_T. Choose any vertex i which has degree of more than two in the tree (there exists at least one if the graph is not a Hamiltonian chain).

Let $$\theta = \{l_{i1}, l_{i2}, \ldots, l_{iP}\} \quad (5\text{-}1)$$

where $l_{ij} \in E_T$, $j = 1, \ldots, p$.

For an arbitrary edge $l_{ij} \in \theta$, define

$$\theta_j = \{l_{ij} | l_{ij} \in \theta\} \notin E_C \quad (5\text{-}2)$$

$$\theta'_j = \{l_{ij} | l_{ij} \in \theta\} \in E_C$$

where E_C is the edge set of an optimal Hamiltonian chain.

Then the edge sets corresponding to the separation S^*_K are:

$$E^K_{i_1} = T_E - \theta_{i_1}$$

$$E^K_{i_2} = T_E \oplus \theta'_{i_1} - \theta_{i_2}$$

$$\vdots \qquad \vdots$$

$$E^K_{i_P} = T_E \oplus \theta'_{i_1} \oplus \theta'_{i_2} \cdots \oplus \theta'_{i_{P-1}} - \theta_{i_P} \quad (5\text{-}3)$$

where T_E is the total edge in the matrix C and the $\oplus$ sign means θ_{i_P} is definitely included in the sub-problem.

We define a shortest spanning tree with this property as a "conditional shortest spanning tree". A detailed discussion of " "conditional shortest spanning tree" will be presented in the next section. We can now state the modified theorem of Garfinkel's Lemma 2 [23] as follows:

Every Hamiltonian chain that is contained in $G^K=(V,E^K)$ is contained in exactly one of the graphs $G^K_{ij}=(V,E^K_{i_P})$, $j=1, \ldots, P$.

Proof:

Let E_C be the edge set of the optimal Hamiltonian chain. Suppose that $E_C \cap \theta'_{i_1} \neq \Phi$, then from (5-3), $E_C \subseteq E^K_{i_1}$ and $E_C \not\subseteq E^K_{i_j}$, $j = 2, \ldots, P$. If $E_C \cap \theta'_{i_1} = \Phi$ and $E_C \cap \theta'_{i_1} \neq \Phi$, the $E_C \not\subseteq E^K_{i_1}$, since $E_C \cap \theta_{i_1} \neq \Phi$. It follows that $E_C \subseteq E^K_{i_2}$ and $E_C \not\subseteq E^K_{i_1}$, $j \neq 2$. By the same reasoning, for $2 \subseteq r \subseteq P$, if $E_C \cap \theta'_{i_j} = \Phi$, $j = 1, \ldots, r-1$, and $E_C \cap \theta'_{i_r} \neq \Phi$, then $E_C \subseteq E^K_{i_r}$ and $E_C \not\subseteq E^K_{i_j}$, $j \neq r$.

This new algorithm for finding a shortest Hamiltonian chain can be summarized as follows:

Step 1: Form a shortest spanning tree for the problem and calculate the lower bound. If the tree forms a Hamiltonian chain, go to Step 2. Otherwise, select vertex i which has maximum number of edges P associated with it. Branch this problem to P sub-problems by using equation (5-3) above. Go to Step 2.

Step 2: If the lower bound for all the sub-problems are greater than the lower bound of the Hamiltonian chain, stop. Otherwise, select the problem with least lower bound and then go to Step 1.

We will use this new procedure to solve the Christofides' example. If we select vertex 2 to form an edge set, the $\theta = \{(2,1),(2,6),(2,5)\}$ and the set S^*_K is determined from:

$$E^K_{2_1} = T_E - (2,1)$$

$$E^K_{2_6} = T_E \oplus (2,1) - (2,6)$$

$$E^K_{2_5} = T_E \oplus (2,1) \oplus (2,6) - (2,5)$$

The tree is shown in Figure 29.

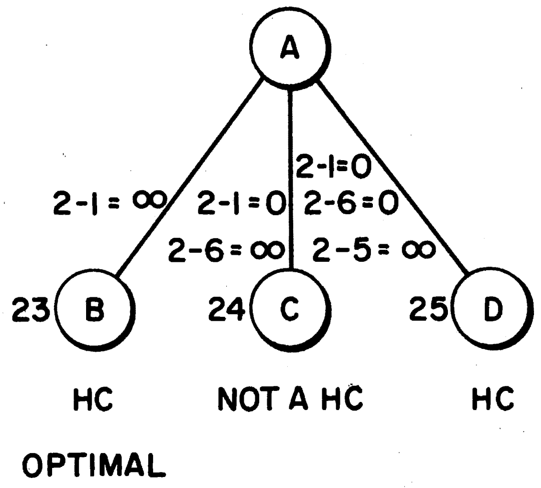


Figure 29. The Decision Tree Search

Although for this small problem the computation work of this procedure is the same as in the Christofides' method, for a large problem it results in a faster convergence to the optimal solution. This conclusion may be verified by comparing the partitioned sub-problems generated from this procedure with those generated from Christofides' method. Using the same combination index, the partitions of the sub-problems are:

Problem B (2-1=∞): This includes the sub-problems with combination index {5, 6, 7, 8}.

Problem C (1-2=0 and 2-6=∞): This includes the sub-problems with combination index {3, 4}.

Problem D (1-2=0 and 2-6=0 and 2-5=∞): This includes the sub-problems with combination index $\{2\}$.

This procedure is therefore more effective than the Christofides' method because all partitions are mutually exclusive.

Description of the Algorithm

A "conditional shortest spanning tree" will now be defined. Given a set of edges E of a graph, and a subset of edges E_S where $E_S \in E$, find a shortest spanning tree which will definitely include E_S in the final solution. The shortest spanning tree resulting is called a "conditional shortest spanning tree". The procedure for finding a "conditional shortest spanning tree" is the same as the procedure of finding shortest spanning tree, except the entries of the cost matrix of all the edges in subset E_S have to be pre-set to zero. Therefore, when Kruskal's method is used, all the edges in E_S will be on the top of ascending list and hence, will always be in the final solution.

We will describe a theorem of Christofides' [10] before presenting the ship routing algorithm.

Theorem: Let $C = [C_{ij}]$ be the link-cost matrix of the original graph G and let K be a large positive number greater than the cost of any Hamiltonian chain. Then the solution of shortest spanning tree with the link-cost matrix C', where:

$$C'_{j,i_1} = C_{j,i_1} + K$$

$$C'_{i_1,j} = C_{i_1,j} + K \qquad \text{for all } j \neq i_1 \text{ or } i_2$$

$$C'_{i_2,j} = C_{i_2,j} + K$$

$$C'_{j,i_2} = C_{j,i_2} + K$$

$$C'_{i,j} = C_{i,j} + 2K \qquad \text{for } i \text{ and } j = i_1 \text{ or } i_2$$

$$C'_{i,j} = C_{i,j} \qquad \text{for all } i,\ j \neq i_1 \text{ or } i_2$$

is the solution of shortest spanning tree in which the degree of vertices i_1 and i_2 is one.

The application of the above theorem will help in forming a shortest Hamiltonian chain which starts from i_1 and ends at i_2 or visa versa. The algorithm for the ship routing problem using graph theoretic approach can now be described. This algorithm combines the following four topics together.

1) The Kruskal's method to form a shortest spanning tree.

2) The "conditional shortest spanning tree" concept described in this section.

3) The Christofides' theorem on a Hamiltonian chain for a specific starting and ending point.

4) The extension of Garfinkel's method of partitioning feasible sets.

The algorithm can be stated as follows:

Step 1: Add a large positive number to the row and the column of P_1 and P_2 in the original cost matrix. Go to Step 2.

Step 2: Use the "conditional shortest spanning tree" concept and use Kruskal's method to form a shortest spanning tree. If the shortest spanning tree is a Hamiltonian chain, go to Step 3. Otherwise, select the vertex of highest degree, and use the extension of Garfinkel's method of partitioning a feasible subset in order to branch the current problem into sub-problems.

Step 3: If the cost of the best shortest Hamiltonian chain generated is less than or equal to the cost of all unsearched sub-problems, stop. Otherwise, select the sub-problem with least cost and then go to Step 2.

The logical flow chart of this algorithm is shown in Figure 30.

Programmed Algorithm

The algorithm for the single-ship through single-prospect problem using the modification of the shortest Hamiltonian chain method was programmed in the FORTRAN IV language. The programmed instructions implement the theory presented in the preceding section.

The algorithm requires a computer memory of approximately 60 K bytes for a 14-line configuration. The time required to obtain the optimal path through a N-line prospect configuration is approximately $(2.67)^N(0.001)$ minutes when executed on the IBM 360/65 computer.

Although the computation time required by this algorithm is larger than the algorithm using the modification of Little's sequential tour-building method, it is found that only this algorithm can solve the problem with an unspecified starting point and ending point. An example of this type of application is presented in Table III.

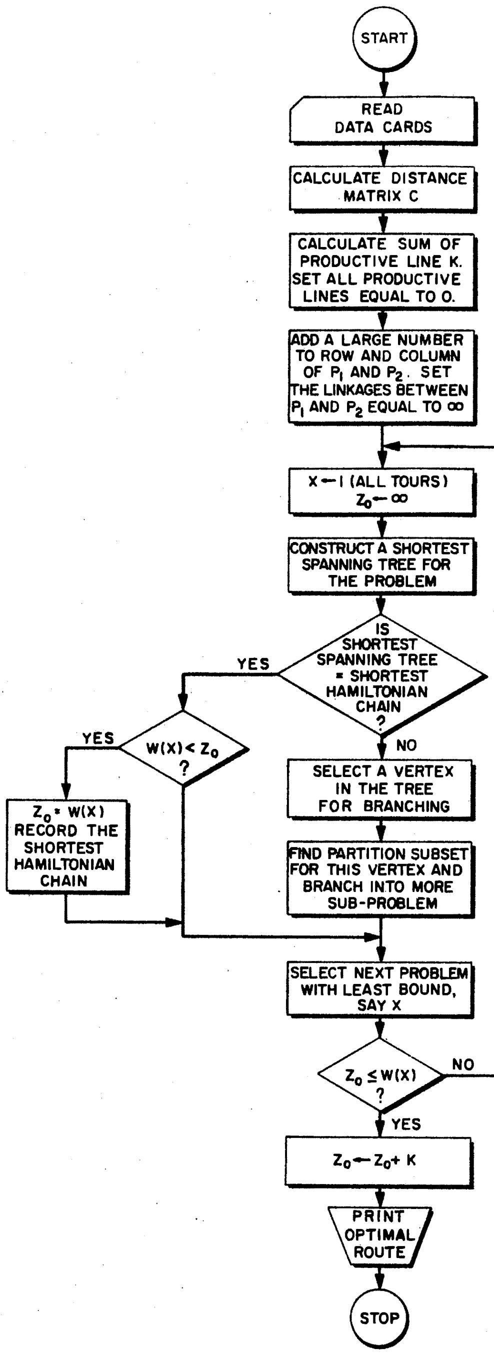

START
READ DATA CARDS
CALCULATE DISTANCE MATRIX C
CALCULATE SUM OF PRODUCTIVE LINE K. SET ALL PRODUCTIVE LINES EQUAL TO 0.
ADD A LARGE NUMBER TO ROW AND COLUMN OF P1 AND P2. SET THE LINKAGES BETWEEN P1 AND P2 EQUAL TO ∞
X ← 1 (ALL TOURS) Z0 ← ∞
CONSTRUCT A SHORTEST SPANNING TREE FOR THE PROBLEM
IS SHORTEST SPANNING TREE = SHORTEST HAMILTONIAN CHAIN ?
YES
NO
W(X) < Z0 ?
YES
Z0 = W(X) RECORD THE SHORTEST HAMILTONIAN CHAIN
SELECT A VERTEX IN THE TREE FOR BRANCHING
FIND PARTITION SUBSET FOR THIS VERTEX AND BRANCH INTO MORE SUB-PROBLEM
SELECT NEXT PROBLEM WITH LEAST BOUND, SAY X
SET UP MATRIX C TO CORRESPOND TO THE PROBLEM X
Z0 ≤ W(X) ?
NO
YES
Z0 ← Z0 + K
PRINT OPTIMAL ROUTE
STOP

Figure 30. The Logic Flow Chart of the Algorithm for Ship Routing Problem Using the Modification of Shortest Hamiltonian Chain Method

Example 6-1: Suppose that a process line manufactures four types of gasoline: racing fuel, premium, regular, and leadfree. In a fuel production cycle, the amount of non-production time (i.e., setup time) depends on the sequence in which these fuels are manufactured. The matrix of setup time S_{ij} might resemble the one shown in Table III. The problem is to find an optimal production sequence for three periods of the fuel production cycle.

TABLE III

THE MATRIX OF SETUP TIMES S_{ij}

		(1)	(2)	(3)	(4)
Racing	(1)	-	30	50	90
Premium	(2)	40	-	20	80
Regular	(3)	30	30	-	60
Leadfree	(4)	20	15	10	-

If the assumption is made that a cyclic plan is always followed, then the problem can be solved by the Little's sequential tour-building method. The optimal sequence for each cycle is the same for each period as shown in Figure 31 and the total cost is 370.

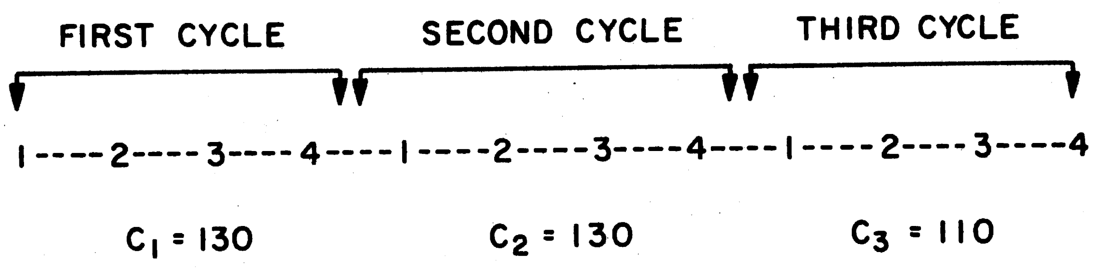


Figure 31. The Optimal Sequence Using Branch and Bound Method

However, it is impractical to assume that the production sequence will be cyclic. It is obvious that if the first sequence of the current cycle maintains the same as the last sequence of the previous cycle, the setup time between these two cycles can be eliminated. To solve the problem under this condition, only the algorithm based on the shortest Hamiltonian chain can be applied. The optimal sequence for each cycle might be different (as shown in Figure 32) and the total cost is 230. Therefore, the shortest Hamiltonian chain method will result in a cost saving of 140 over branch and bound method.

The comparison of execution times and storage requirements between this algorithm and the algorithms based on other approaches will be discussed in Chapter VI.

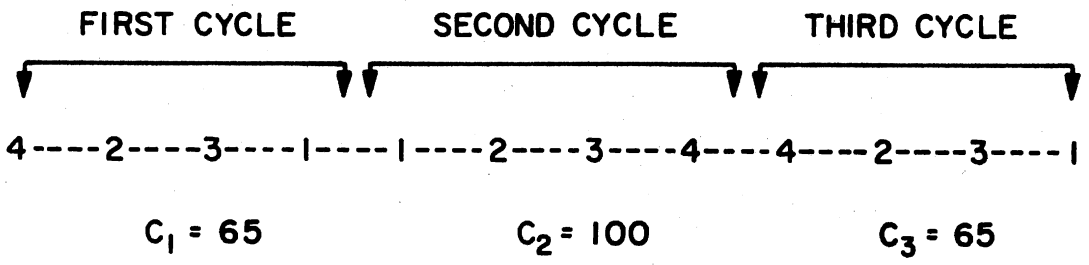


Figure 32. The Optimal Sequence Using Shortest Hamiltonian Chain Method

CHAPTER VI

SUMMARY OF ALGORITHMS FOR THE SINGLE-SHIP THROUGH SINGLE-PROSPECT ROUTING PROBLEM

Summary of Algorithms

In the research work of Phase I, four algorithms were developed using different approaches to solve the single-ship through single-prospect problem. A total of seven separate computer programs implementing the theories developed, were coded into the FORTRAN IV language on the IBM 360/65 computer.

Table IV shows: 1) approximate computational times, 2) memory storage requirements, 3) maximum number of lines in programmed algorithm, and 4) total number of data sets tested. The most effective algorithm developed to solve the single-ship through single-prospect problem is the algorithm based upon the modification of Little's sequential tour-building method. The computer program for this algorithm requires only 60 K bytes to solve a 14-line configuration problem while the Willard's algorithm [66] takes 150 K bytes to solve a 10-line configuration problem. The computation time for this algorithm is only slightly higher than Willard's.

A drawback of this algorithm is that for a small portion of test problems with lines numbering over 13, the algorithm might converge to optimal solution very slowly. This is a common obstacle

TABLE IV

THE COMPARISON OF PERFORMANCE FOR VARIOUS ALGORITHMS

Method \ Comparison Items			Approx.* Computat. Time (minutes)	Memory Storage Requirem. (K Bytes)	Maximum No. Line In Porg. Algorithm	Total Number of Data Set Tested
Dynamic Programming Approach	Conventional Method(66)		$(1.84)^N$ (0.001)	152	8	6
	Recursive Search Method		$(2.35)^N$ (0.001)	62	8	6
Branch And Bound Approach	Modification of Little's Branch & Bound Method	Strategy (1) and Version (1)	$(2.09)^N$ (0.001)	62	14	12
		Strategy (2) and Version (1)	$(2.05)^N$ (0.001)	92	14	12
		Strategy (2) and Version (2)	$(2.07)^N$ (0.001)	62	14	12
	Modification of Subtour Elimination Method		$(2.92)^N$ (0.001)	60	14	5
Graphic Theory Approach- Modification of Shortest Hamiltonian Method			$(2.67)^N$ (0.001)	62	14	10

*The approximate computation times for N lines is an average of execution time of test problems which excluded the problems that converge to optimal solution very slowly.

faced by optimal seeking procedures for the combinatorial problems having large numbers of entries. To overcome this potential difficulty, an effective strategy for obtaining a heuristic solution has been developed and incorporated into this algorithm. The detail of this strategy is described in the next section.

Modification of Optimizing Algorithm With Heuristic Solution Features

As mentioned previously, the difficulty faced by optimal seeking procedures for large scale combinatorial problems is that computation time increases exponentially with the number of entries N. This fact was experienced when the algorithms developed in the previous chapters for the single-ship through single-prospect routing problem were tested. For this reason, a heuristic solution approach is studied in this dissertation. The purpose is to develop an effective heuristic solution strategy to incorporate into the optimizing algorithm so that a near-optimal solution can be obtained whenever the seeking of optimal solution is difficult.

Bounds of the Problem

The quality of the lower bound is a vital factor in determining the effectiveness of the branch and bound algorithm developed in the previous two chapters. A good quality lower bound is also useful when establishing a reference point against which one can compare the results of heuristic solution methods.

Christofides [21] suggest three ways of calculating lower bounds for traveling salesman problems. All are easy to calculate, and the computation time is negligible when compared with the total solution time of the traveling salesman problem. These three ways can be described briefly as follows:

1) The Shortest Spanning Tree (SST)

The traveling salesman problem can be represented by a graph of N nodes and N(N-1) possible links which completely interconnect these nodes. A spanning tree of such a graph is any set of (N-1) links that connect all of the N nodes. If a link is removed from a Hamiltonian circuit, then the graph becomes a Hamiltonian chain. A Hamiltonian chain is a particular case of a spanning tree and is thus limited from below by the length of the SST. Instead of removing any link from Hamiltonian circuit, Christofides suggested removing the longest one which will yield a better bound for the traveling salesman problem. This implies the following equation for the lower bound to the optimal traveling salesman tour.

$$B_{SST} = C_{SST} + \max_{j} (d_{j2})$$

where C_{SST} is the cost of the SST and d_{j2} is the cost between the jth city and the second nearest of its neighboring cities.

2) The Assignment Problem (AP)

The traveling salesman problem can be treated as a classical problem with the extra requirement that the solution of assignment problem must be a tour. Thus the solution to the assignment problem which does not necessarily yield a valid tour is a lower bound for the optimal traveling salesman tour.

3) The Sum of the Shortest Link (SL)

In a graph that depicts a traveling salesman tour, two links emanate from each node. Thus, a valid lower bound for the cost of the traveling salesman tour is the quantity

$$B_{SL} = 1/2 \sum_{j=1}^{N} (d_{j1} + d_{j2})$$

where d_{j1} and d_{j2} are the shortest links and the next shortest links that can emanate from city j, respectively. The factor 1/2 is necessary because each city is considered twice as "go-to" and "come-from".

Various experiments have been performed to test the lower bounds suggested above by Christofides. Ten traveling salesman problems with 10, 20, etc., up to 100 cities were randomly generated. The problems were solved by 3-optimal method (a tour is defined to be 3-optimal if no improvement can be obtained by replacing any r of its links by any other set of r links). Bounds were calculated for the problem without any attempts to improve them and the results as reported by Christofides are shown in Figure 33.

The bounds from the SST are seen to be the best for all ten problems with an average value of about ten per cent below the costs of the conjectured optimal tour. Next best are the bounds B_{SL} and B_{AP}, with average values of 14 per cent and 19 per cent, respectively, below the optimal tour costs. Based upon this result, the bounds generated by the SST concept which are the tightest among the three lower bounds, will be used to develop an effective heuristic solution strategy that can be incorporated into the optimizing algorithm.

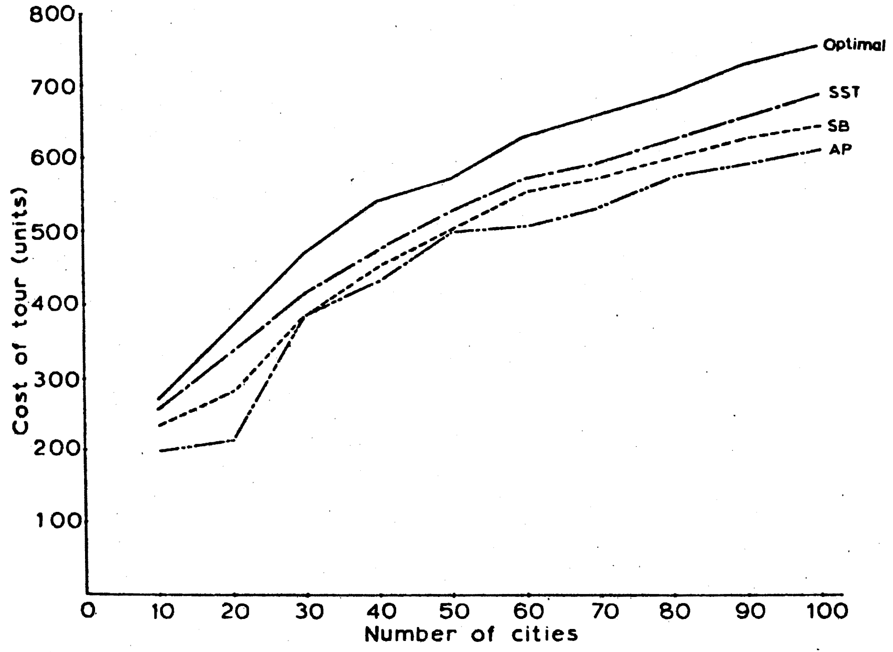


Figure 33. A Comparison of Lower Bounds in Ten Traveling Salesman Problems from Christofides' [21] Paper

Description of Optimizing Algorithm With Heuristic Solution Feature

The optimizing algorithm with heuristic solution feature developed in this section uses the lower bound generated from the shortest spanning tree problem to stop the optimal seeking procedure and obtain a near-optimal solution when the computation time exceeds a specified limit. The algorithm can be stated as follows:

Step 1: Pre-determine an execution time limit of the program, Δs. The limit Δs will be used to stop the optimal seeking procedure of the algorithm if the difference between the best cost yielded and the cost of shortest spanning tree (SST) is less than or equal to Δs. Calculate the cost of SST of the problem, C_B.

Step 2: Check the total execution time against the execution time limit of the problem. If the total execution time is less than the execution time limit, go to Step 4. Otherwise, go to Step 3.

Step 3: If the best solution yielded C_T, satisfy the following condition:

$$C_T - C_b \leq \Delta s$$

then indicate the last solution is an approximation and stop. Otherwise, go to Step 4.

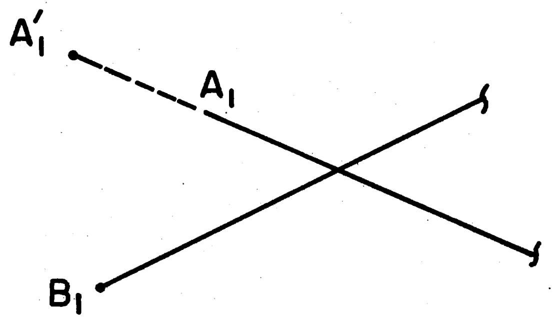
Step 4: Use the optimizing algorithm based upon the modification of Little's sequential tour-building method to search each sub-problem. If the cost of all unsearched sub-problems is greater than or equal to the cost of the best solution yielded so far, stop. Otherwise, find the next problem to be searched and to to Step 2.

The optimizing algorithm with the heuristic solution feature for the single-ship through single-prospect routing problem was programmed in the FORTRAN IV language. The algorithm requires the same computer storage of version (2)-strategy(2) of the algorithm using the modification of Little's sequential tour-building method. Several prospect path-configurations have been used to test the computation performance of the algorithm. It was found that the algorithm will produce a good near-optimal solution whenever the conditions prohibit convergence to the true optimal solution. This algorithm is used in the further research of phase II.

Extension of the Algorithm

The programmed algorithm developed in the previous section will select the optimal path or near-optimal path of the problem based upon the distance matrix calculated from the coordinates of all seismic lines and starting and ending ports. However, in practice, the operational aspects of the ship which collects the data on seismic lines must be taken into consideration before an initial penalty matrix is generated.

When the recording ship is in port, the seismic cable is wound on a reel aboard the ship. When collecting data on a line, this cable must be laid out in the water and towed by the ship. The axis of the cable must be aligned with the line being traversed as data is being gathered. In order to cover the entire seismic line, a ship must travel to a further position from the end point of the line so that when the ship arrives at the end point, the cable will be in the correct position. To lay out or pull in a cable requires approximately one to three hours, depending on the cable length and the mechanical equipment installed on the ship. With the cable aboard the ship, an average ship can travel approximately ten to fifteen knots per hour. If the cable is towed, the ship's speed is reduced approximately five knots per hour because of the severe drag. Since there is a difference in speeds with the cable in or not, a decision must be made on whether to leave the cable in the water and change line at the slower speed or pull in the cable and travel to the next line faster. A break-even distance where it is equally advantageous to both alternatives can be calculated and used to make a favorable decision. To take into consideration the operational aspects described above, Willard [66] suggested that a penalty matrix representing travel time is more appropriate than distance. Willard's method of generating a penalty matrix of travel time is summarized as follows:



Define: T_{in} = time required to pull cable in.

T_{out} = time required to lay cable out.

S_{in} = speed with cable in.

S_{out} = speed with cable out.

L = the length of cable.

The calculation of the penalty for changing lines from B_1 to A_1 is:

1) Calculate new co-ordinates A_1' for A_1

Distance from A_1 to $A_1' = 2\,L$

2) Calculate distance D from B_1 to A_1'

3) Calculate break-even travel distance D* by

$$D^* = (T_{in} + T_{out})/[(1/S_{out}) - (1/S_{in})]$$

4) If $D < D^*$ Time travel from B_1 to $A_1 = D/S_{out}$

$D \geq D^*$ Time travel from B_1 to $A_1 = [D/S_{in}] + T_{in} + T_{out}$

also Time travel from P_1 to $A_1' = [(\text{distance from } P_1 \text{ to } A_1')/S_{in}] + T_{out}$

Time travel from A_1 to $P_2 = [(\text{distance from } A_1 \text{ to } P_2)/S_{in}] + T_{in}$

The capability of routing the ship by time suggested by Willard is incorporated into the optimizing algorithm with the heuristic solution feature described in the previous section. This additional information will aid management in effectively utilizing the ship.

CHAPTER VII

ALGORITHMS FOR LARGE SCALE SHIP ROUTING PROBLEM

In today's marine seismic mapping operations, situations that require a ship to cover multi-prospects or multi-ships to cover multi-prospects are common. The number of feasible paths that can be selected is much larger than the single-ship through single-prospect problem. It is economically desirable that an effective algorithm be developed for such large scale mapping operations. For this reason, the Phase I research work of the previous chapters is extended in this chapter which considers the study and development of algorithms for the single-ship through multi-prospects problem and the multi-ships through multi-prospects problem. The development of algorithms in this chapter is concentrated on heuristic solution approaches.

The research work of Phase II is accomplished in two parts: 1) the development of algorithms and 2) the development of a multi-purposes computer program.

The Development of Algorithms

Algorithm for the Single-Ship Through Multi-Prospects Problem

To develop an algorithm for the single-ship through multi-prospects problem, the location of each prospect and the lengths of the seismic lines in each prospect have to be carefully examined. Figure 34 shows three examples with different kinds of layout. In Example 1, the locations of the propsects are apart from each other. The average length of the seismic lines in each prospect is comparatively smaller than the distance between the prospects. In Example 2, the locations of the prospects are close to each other, and the average lengths of the seismic lines in each prospect are also very similar to each other. In Example 3, the locations of the prospects are also close to each other, but the average length of the seismic lines of one prospect is comparatively larger than the average line lengths of the other two prospects.

Example 1 is a representative layout of current marine seismic mapping operations. Each prospect is far away from the other prospects and the average lengths of the seismic lines in each prospect vary. Therefore, the development of the algorithm for solving the single-ship through multi-prospects problems will be concentrated on this type of problem.

To route a ship through all the prospects like this type, the best sequence of the prospects which start at P_1 and end at P_2 can be determined first before the routing process for each prospect begins. The extent of variation from true optimal solution resulted from the predetermined sequence and can be minimized since the average length of the seismic lines in each prospect is much smaller than the distances between the prospects.

To determine the best sequence of prospects, a center for each prospect is first calculated by averaging the coordinates of the end points of each line within a prospect. A second point is then created

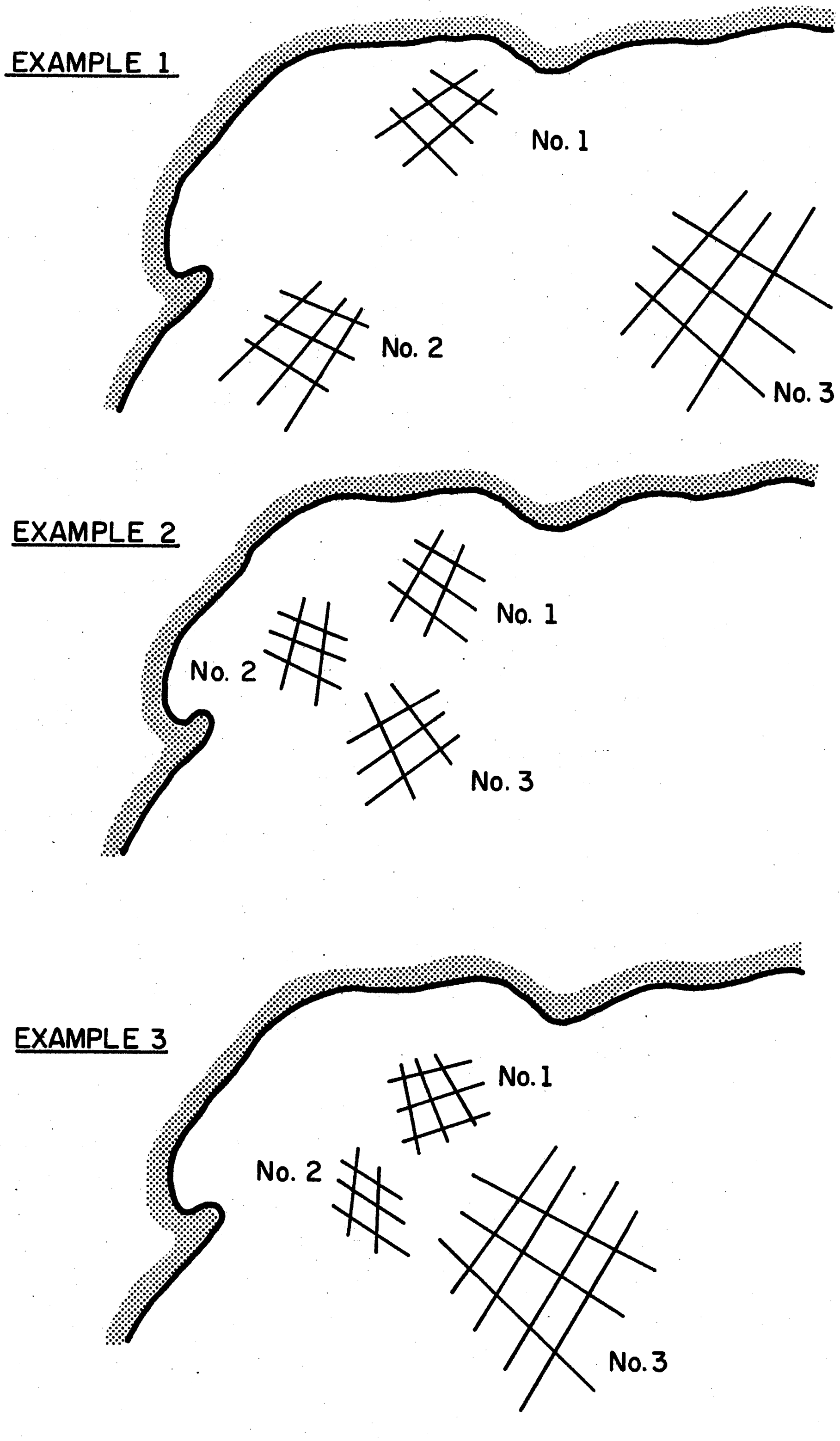


Figure 34. Examples of the Single-Ship Through Multi-Prospects Problem

at a position near the previous point and a line between these two points is drawn. With this kind of formulation, the problem of routing the sequence of four prospects is now the problem of routing a prospect with four seismic lines, and the algorithm developed in Phase I can be applied.

To route a single ship through N prospects, the procedure is based upon the idea that the best sequence can be pre-determined as above. The algorithm developed for the single-ship through single-prospect problem can then be repeatedly applied for N times until the total problem is solved. However, the starting point and the ending point for each iteration of a prospect routing is different and has to be determined before application of the algorithm. For the first prospect in the best sequence, the starting point is equal to the starting port in the total problem. For the last prospect in the best sequence, the ending point is equal to the ending port in the total problem. For the prospect in the middle of the best sequence, the starting point is equal to the ending point of last seismic line of previous prospect routing and the ending point is equal to the center of the next prospect in the best sequence.

The algorithm developed for solving the single-ship through multi-prospects problem can be summarized as follows:

Step 1: Determine the best sequence of all N prospects for the ship to be routed by

1) Calculate the center of each prospect.
2) At each prospect, create a second point which varies only slightly from the center of each prospect.

3) Join the line between the point in 1) and 2).

4) Apply the best algorithm developed for the single-ship through single-prospect problem to obtain a best sequence of N prospects.

Step 2: If all N prospects have been routed, stop. Otherwise, determine the starting point and the ending point for the i^{th} prospect in the sequence to be routed by

1) If $i = 1$, the starting point is equal to P_1. Otherwise, the starting point is equal to the ending point of $(i - 1)^{th}$ prospect in the sequence.

2) If $i = N$, the ending point is equal to P_2. Otherwise, the ending point is equal to the center of the $(i + 1)^{th}$ prospect in the sequence. Go to Step 3.

Step 3: Apply the best algorithm developed for the single-ship through single-prospect problem to obtain optimal solution for the prospect, if possible. Obtain a heuristic solution for the prospect if the optimal solution is not able to reach. Go to Step 2.

For the type of problem in Example 2 and Example 3 of Figure 34, the application of the procedure which pre-determines the sequence of prospects of the problem before routing must be done very carefully. It is possible that the sequence which would result in a true optimal solution for the problem is different from the best sequence determined by using the center of prospects. For this reason, it is strongly suggested that if there exists a solution which appears to be better instinctively, this solution should be routed and the result compared to the result obtained from that procedure so that the most favorable decision can be made.

The computer program for the above procedure, including the option of user specified route, is a part of the multi-purposes computer program which will be described in the next section of this chapter.

Algorithm for Multi-Ships Through Multi-Prospects Problem

In routing multi-ships through multi-prospects, the characteristics of each ship, such as tonnages, capacity, equipment, etc., might be different. In addition, the locations of the prospects might have some restrictions to certain ships due to the depth of ocean floor or the current of the sea at a particular time. For these reasons, it is not practical to seek an optimal procedure which will include the above described conditions for multi-ships through multi-prospects problem. In this dissertation, a more flexible procedure which enables a user to select certain prospects to be routed by a certain ship is developed.

The algorithm developed for solving the M ships through N prospects problem can be described as follows:

1) A user will assign a selection of the N prospects to be routed by a particular ship.
2) The algorithm developed for solving the single-ship through multi-prospects problem is used to route each assignment for all ships.
3) If the result obtained is unsatisfactory, or if more data are needed for comparison purposes, a user can revise the

assignments and implement the process again.

The computer program for the above procedure is also a part of the multi-purposes computer program which will be described in the following section.

The Development of a Multi-Purposes Computer Program

A well documented multi-purposes computer program which includes the extension of the algorithm of Chapter VI and the procedures discussed in the previous two sections of this chapter is given in this dissertation. This program provides the following options for a user:

1) To route the single-ship through single-prospect problem
    a) either by distance
    b) or by time
2) To route the single-ship through multi-prospects problem
    a) either by distance
    b) or by time
3) To route the multi-ships through multi-prospects problem
    a) either by distance
    b) or by time
4) The program has a built-in capability for routing ships through any user assigned path. The purpose of this capability is to provide a user a way of comparing the result obtained from his assigned route and the result obtained from the algorithm.
5) The program has a built-in capability for overriding any entries in the distance matrix.

The computer program with the above options was coded into the FORTRAN IV language. The program takes approximately 98 K bytes of computer memory storage and is very fast in computation time. The program can solve the following sizes of problems:

1) It can route an unlimited number of ships.
2) For each ship, it can solve five maximum prospects with each prospect having up to 14 lines.

The capability of solving a larger size of problem can be achieved very easily by increasing the dimensions in the computer program. A logic flow chart of this multi-purposes computer program is shown in Figure 35. A list of the source program with necessary documentation is included in the Appendix.

A representative example of a seismic configuration with three prospects is shown in Figure 36. The above described multi-purposes computer program is used to route this example by both distance and time. In the case of routing the ship by distance, only the identification of the route and the total distance of this route is printed. The output indicates the solution type; i.e., optimal solution, near-optimal solution, or solution of user assigned route. In addition, the identification of the starting point and the ending point of each prospect routing is indicated in the path information.

In the case of routing the ship by time, the output also includes the position of the cable during each line change that minimizes the line change time. Mileages and times are printed for both the individual path segment and for the total prospect. The total times and distances are divided into productive and non-productive portions. Productive time is the time when the crew is actually collecting the seismic data,

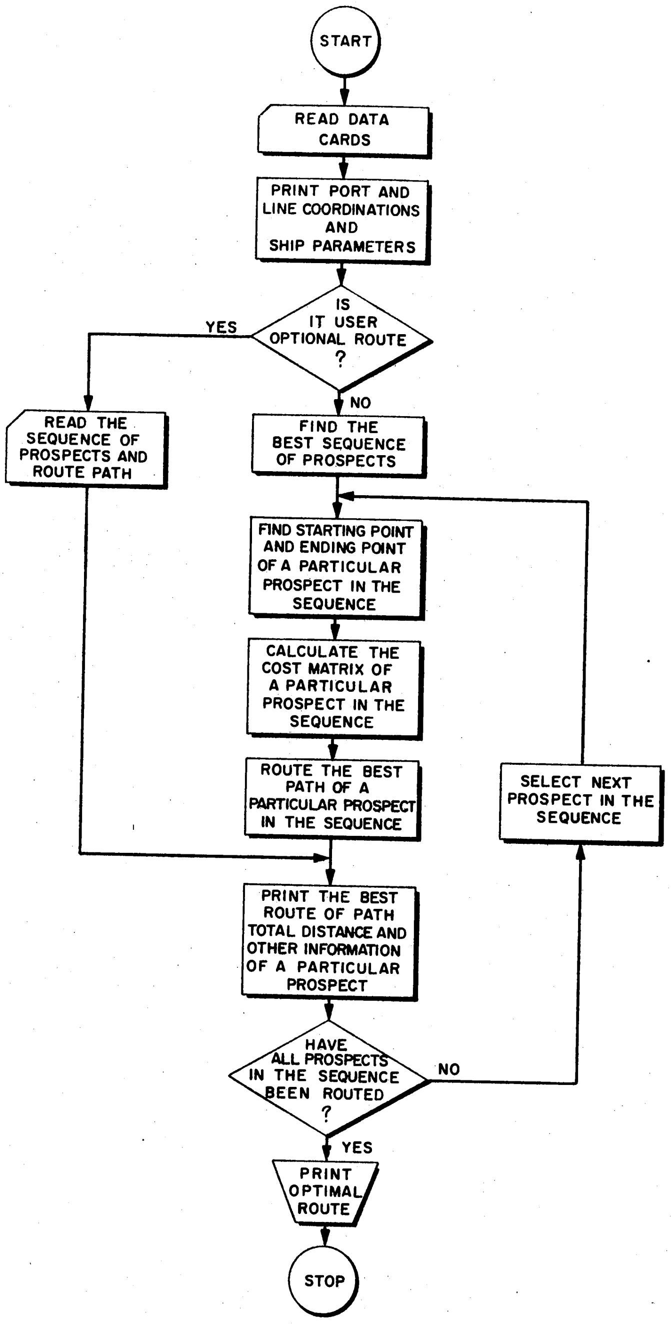

START
READ DATA CARDS
PRINT PORT AND LINE COORDINATIONS AND SHIP PARAMETERS
IS IT USER OPTIONAL ROUTE ?
YES
NO
READ THE SEQUENCE OF PROSPECTS AND ROUTE PATH
FIND THE BEST SEQUENCE OF PROSPECTS
FIND STARTING POINT AND ENDING POINT OF A PARTICULAR PROSPECT IN THE SEQUENCE
CALCULATE THE COST MATRIX OF A PARTICULAR PROSPECT IN THE SEQUENCE
ROUTE THE BEST PATH OF A PARTICULAR PROSPECT IN THE SEQUENCE
SELECT NEXT PROSPECT IN THE SEQUENCE
PRINT THE BEST ROUTE OF PATH TOTAL DISTANCE AND OTHER INFORMATION OF A PARTICULAR PROSPECT
HAVE ALL PROSPECTS IN THE SEQUENCE BEEN ROUTED ?
NO
YES
PRINT OPTIMAL ROUTE
STOP

Figure 35. The Logic Flow Chart of the Algorithm for Multi-Ship Through Multi-Prospects Routing Problem

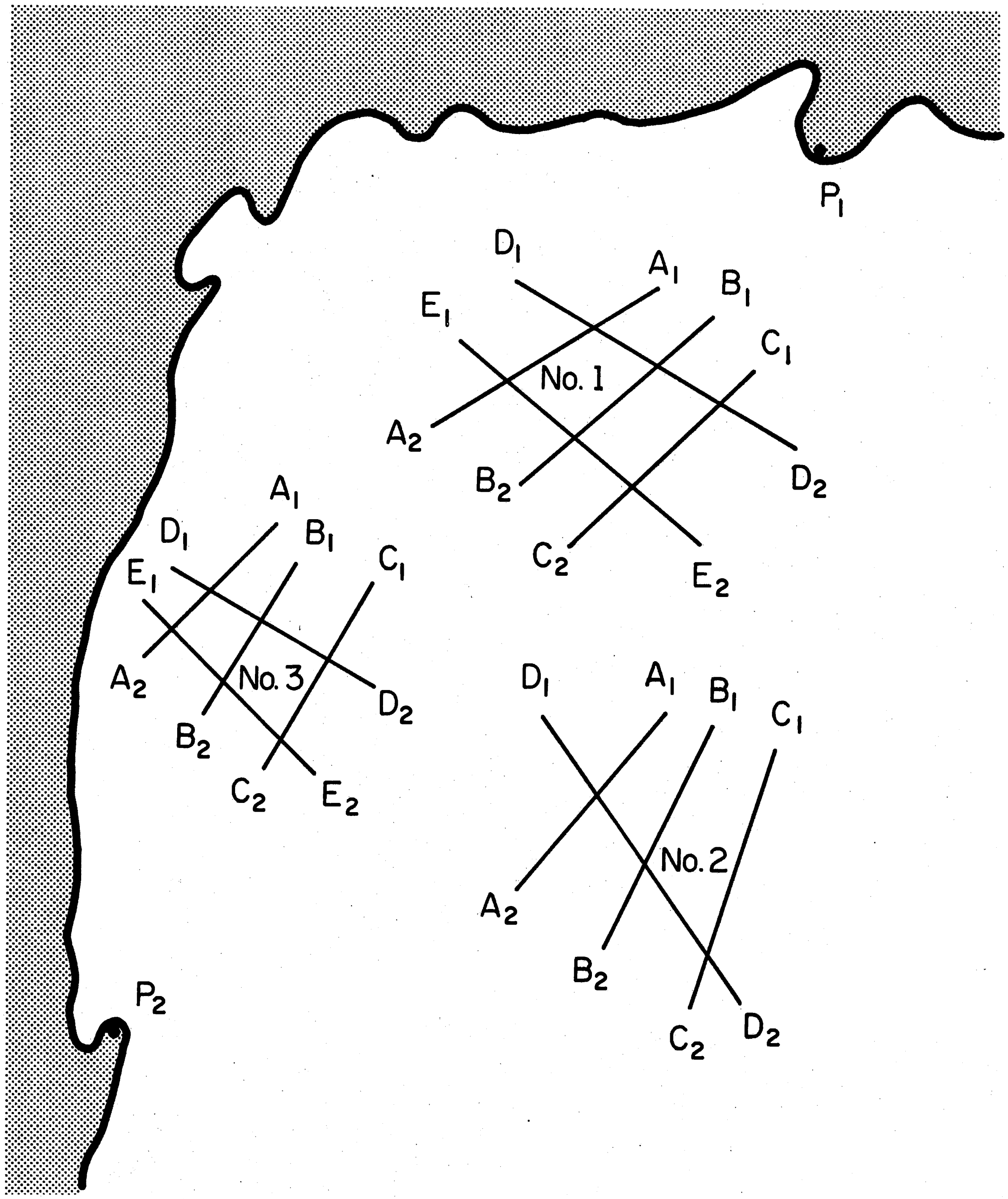


Figure 36. An Example of One Ship Through Three Prospect Problem

while non-productive time is the elapsed time going both to and from prospects and changing lines within prospects. Although the identification of the route is the information that is of primary importance, the additional information will aid management in effectively utilizing the ship. The output of this example, routing by time, is shown in Figure 37 through Figure 40.

```
*************************************************
*                                               *
*                                               *
* 1-SHIPS THROUGH 3-PROSPECTS ROUTING PROBLEM   *
*                                               *
*                                               *
*************************************************
                    SHIP NO:  1
*************************************************
*                                               *
*                                               *
*                SHIP PARAMETERS                *
*                                               *
*                                               *
*           CABLE LENGTH = 7874 FEET            *
*           SCALE =  1.00 UNITS/MILE            *
*                                               *
*       SHIP SPEED (CABLE OUT) =  6.00 MPH      *
*       SHIP SPEED (CABLE IN)  = 15.00 MPH      *
*                                               *
* CABLE HANDLING TIME (LAY OUT) = 1.25 HOURS    *
* CABLE HANDLING TIME (PULL IN) = 1.75 HOURS    *
*                                               *
*                                               *
*************************************************
*                                               *
*                                               *
*               PORT CO-ORDINATES               *
*                                               *
*                                               *
*               _X1_  _Y1_      _X2_  _Y2_      *
*                                               *
*     P          48.   89.        0.   28.      *
*                                               *
*                                               *
*************************************************
*                                               *
*                                               *
*               PROSPECT NO:  1                 *
*                                               *
*                                               *
*               _X1_  _Y1_      _X2_  _Y2_      *
*                                               *
*     A          38.   80.       22.   70.      *
*     B          42.   78.       28.   66.      *
*     C          44.   74.       32.   62.      *
*     D          28.   80.       48.   68.      *
*     E          24.   76.       40.   62.      *
*                                               *
*                                               *
*************************************************
*                                               *
*                                               *
*               PROSPECT NO:  2                 *
*                                               *
*                                               *
*               _X1_  _Y1_      _X2_  _Y2_      *
*                                               *
*     A          38.   50.       28.   38.      *
*     B          42.   50.       34.   34.      *
*     C          46.   48.       40.   30.      *
*     D          30.   50.       44.   30.      *
*                                               *
*                                               *
*************************************************
*                                               *
*                                               *
*               PROSPECT NO:  3                 *
*                                               *
*                                               *
*               _X1_  _Y1_      _X2_  _Y2_      *
*                                               *
*     A          10.   62.        2.   54.      *
*     B          12.   60.        6.   50.      *
*     C          18.   60.       10.   46.      *
*     D           4.   60.       18.   52.      *
*     E           2.   58.       14.   46.      *
*                                               *
*                                               *
*************************************************
```

Figure 37. Output Data-1

```
                      PENALTY MATRIX FOR PROSPECT NO: 1

       P2     A1     A2     B1     B2     C1     C2     D1     D2     E1     E2
  ****************************************************************************

P1 *   1      1      3      1      3      2      3      2      2      3      3

A1 *   4   ****      3      1      3      1      4      2      3      3      3

A2 *   3      3   ****      4      1      4      2      2      5      1      4

B1 *   4      1      4   ****      3      1      4      3      2      3      3

B2 *   3      3      1      3   ****      3      1      3      4      2      2

C1 *   3      1      4      1      3   ****      3      3      2      4      2

C2 *   3      3      2      4      1      3   ****      3      3      3      2

D1 *   4      2      2      3      3      3      3   ****      4      1      4

C2 *   3      2      5      2      4      1      3      4   ****      5      2

E1 *   4      3      1      3      2      4      3      1      5   ****      4

E2 *   3      3      4      3      2      2      2      4      2      4   ****
NCTE: P2 IS AN ARTIFICAL ENDING PORT.
```

```
                    FRCSPECT NO:  1
 ***************************************************
 *                                                 *
 *                                                 *
 *          OPTIMUM PATH INFCRMATION               *
 *                                                 *
 *                                                 *
 *   FRCM  TO      CABLE       MILES      HOURS    *
 *                                                 *
 *    P1 - B1        IN         12.         1.     *
 *                                                 *
 *    B1 - B2       OUT         18.         3.     *
 *                                                 *
 *    B2 - A2       CUT         11.         1.     *
 *                                                 *
 *    A2 - A1       OUT         19.         3.     *
 *                                                 *
 *    A1 - C1       CUT         11.         1.     *
 *                                                 *
 *    C1 - C2       CUT         17.         3.     *
 *                                                 *
 *    C2 - E2       CUT         12.         2.     *
 *                                                 *
 *    E2 - E1       OUT         21.         4.     *
 *                                                 *
 *    E1 - D1       CUT          8.         1.     *
 *                                                 *
 *    D1 - D2       OUT         23.         4.     *
 *                                                 *
 *    D2 - #2        IN         29.         3.     *
 *                                                 *
 *                                                 *
 ***************************************************
 *                                                 *
 *                                                 *
 *                           HOURS      MILES      *
 *                                                 *
 *      PRCDUCTIVE             16.        99.      *
 *                                                 *
 *      NON-PRODUCTIVE      ____6.     ___54.      *
 *                                                 *
 *      TOTAL                  22.       153.      *
 *                                                 *
 *                                                 *
 ***************************************************
```

Figure 38. Output Data-2

```
                PENALTY MATRIX FOR PROSPECT NO: 2


        P2    A1    A2    B1    B2    C1    C2    D1    D2
     ********************************************************

P1 *    1     2     3     2     3     2     3     2     3

A1 *    3  ****     2     0     2     1     3     1     3

A2 *    3     2  ****     3     1     3     2     2     3

B1 *    3     0     3  ****     3     0     3     2     3

B2 *    3     2     1     3  ****     3     1     2     1

C1 *    4     1     3     0     3  ****     3     2     3

C2 *    4     3     2     3     1     3  ****     3     0

D1 *    3     1     2     2     2     2     3  ****     4

D2 *    4     3     3     3     1     3     0     4  ****

NOTE: P1 IS AN ARTIFICAL STARTING PORT AND P2 IS AN ARTIFICAL ENDING PORT.

                      PROSPECT NO:  2
     **********************************************
     *                                            *
     *                                            *
     *           OPTIMUM PATH INFORMATION         *
     *                                            *
     *                                            *
     *   FROM  TO     CABLE     MILES      HOURS  *
     *                                            *
     *    #1 - A1       IN       21.        2.    *
     *                                            *
     *    A1 - A2      OUT       16.        3.    *
     *                                            *
     *    A2 - B2      CUT        7.        1.    *
     *                                            *
     *    B2 - B1      OUT       18.        3.    *
     *                                            *
     *    B1 - C1      OUT        4.        0.    *
     *                                            *
     *    C1 - C2      CUT       19.        3.    *
     *                                            *
     *    C2 - D2      OUT        4.        0.    *
     *                                            *
     *    D2 - D1      OUT       24.        4.    *
     *                                            *
     *    D1 - #3       IN       21.        3.    *
     *                                            *
     *                                            *
     **********************************************
     *                                            *
     *                                            *
     *                         HOURS    MILES     *
     *                                            *
     *     PRODUCTIVE           13.      77.      *
     *                                            *
     *     NON-PRODUCTIVE     ___3.    __36.      *
     *                                            *
     *     TOTAL                16.     113.      *
     *                                            *
     *                                            *
     **********************************************
```

Figure 39. Output Data-3

```
                    PENALTY MATRIX FOR PROSPECT NO: 3

        P2    A1    A2    B1    B2    C1    C2    D1    D2    E1    E2
     ******************************************************************

P1 *     1     2     3     2     2     2     2     3     2     3     2

A1 *     4  ****     1     0     2     1     2     1     2     1     2

A2 *     3     1  ****     2     1     2     1     1     2     0     2

B1 *     4     0     2  ****     2     1     2     1     1     1     2

B2 *     3     2     1     2  ****     2     1     1     2     1     1

C1 *     4     1     2     1     2  ****     2     2     1     2     2

C2 *     3     2     1     2     1     2  ****     2     1     2     0

D1 *     3     1     1     1     1     2     2  ****     2     0     2

D2 *     3     2     2     1     2     1     1     2  ****     2     1

E1 *     3     1     0     1     1     2     2     0     2  ****     2

E2 *     3     2     2     2     1     2     0     2     1     2  ****
NOTE: P1 IS AN ARTIFICAL STARTING PORT.
```

```
                         PROSPECT NO:  3
          ***********************************************
          *                                             *
          *          OPTIMUM PATH INFORMATION           *
          *                                             *
          *  FROM  TO     CABLE     MILES      HOURS    *
          *                                             *
          *   #2 - B2       IN       24.         2.     *
          *                                             *
          *   B2 - B1      OUT       12.         2.     *
          *                                             *
          *   B1 - A1      OUT        3.         0.     *
          *                                             *
          *   A1 - A2      OUT       11.         2.     *
          *                                             *
          *   A2 - E1      OUT        4.         0.     *
          *                                             *
          *   E1 - E2      OUT       17.         3.     *
          *                                             *
          *   E2 - C2      OUT        4.         0.     *
          *                                             *
          *   C2 - C1      OUT       16.         3.     *
          *                                             *
          *   C1 - D2      OUT        8.         1.     *
          *                                             *
          *   D2 - D1      OUT       16.         3.     *
          *                                             *
          *   D1 - P2       IN       32.         3.     *
          *                                             *
          ***********************************************
          *                                             *
          *                       HOURS    MILES        *
          *                                             *
          *     PRODUCTIVE         12.      72.         *
          *                                             *
          *     NON-PRODUCTIVE    __6.    __75.         *
          *                                             *
          *     TOTAL              18.     147.         *
          *                                             *
          ***********************************************
```

Figure 40. Output Data-4

CHAPTER VIII

CONCLUSIONS AND RECOMMENDATIONS

This chapter includes a summary of how the research objectives set forth in Chapter I were accomplished, a summary of the results, and suggestions for future research.

Conclusions

The literature which relates to this research was surveyed extensively and described in Chapter II of this dissertation. The research work of this dissertation was done in two phases. Phase I research work concentrated on the development of algorithms, to select the minimum cost route for the single-ship through single-prospect problem that improve a previous algorithm developed by Willard (66) upon execution times and/or memory storage requirements. Phase II research work concentrated on the extension of the results in Phase I to the development of an algorithm for large scale seismic mapping operations which includes the single-ship through multi-prospects problem and the multi-ships through multi-prospects problem.

The research work of Phase I consisted of three sub-objectives. The first sub-objective of the Phase I research work was to study the applicability of the dynamic programming approach and develop an improved algorithm for solving the single-ship through single-prospect problem. Chapter III of this dissertation described the development of a new algorithm using a recursive search dynamic programming method. This method starts with an initial feasible solution. At each stage, a state variable one at a time is generated when it is needed. The resulting solution is used to improve the initial solution until the optimum solution is reached. This procedure is different from the conventional dynamic programming method which is required to determine the optimal value of each decision variable for each feasible input state. The comparison of this new algorithm with the Willard's algorithm [66] shows that the computer memory storage requirement can be reduced with a trade off of increasing computation time at about the same rate.

The second sub-objective of the Phase I research was to study the applicability of the branch and bound approach, and to develop an algorithm for solving the single-ship through single-prospect problem. Two methods, Little's sequential tour-building method and Eastman's subtour elimination method, were studied. The development of two new algorithms based upon modifications of the above two methods are described in Chapter IV of this dissertation. The algorithm using the modification of Eastman's subtour elimination was shown to be ineffective when applied to this particular problem. However, the algorithm using the modification of Little's sequential tour-building method was shown to be very effective in computation time and required only a small amount of computer memory storage.

The third sub-objective of the Phase I research was to study the applicability of the graph theoretic approach and to develop an algorithm for solving the single-ship through single-prospect problem. Chapter V of this dissertation described the development of a new algorithm based upon a modification of the shortest Hamiltonian chain method. Although the algorithm developed is ineffective when compared to the other algorithms, it does have other important applications such as the machine sequence scheduling problem with unknown starting and ending entries.

The comparison of execution time and storage requirement for all algorithms developed in the Phase I research work is described in Chapter VI of this dissertation. Some modifications of the optimizing algorithm to include heuristic solution features and other extensions to include operational aspects of seismic recording ships are also described.

The objective of the Phase II research was to extend the results of Phase I to develop a new algorithm for solving the single-ship through multi-prospects problem and the multi-ships through multi-prospects problem. Chapter VII of this dissertation describes the development of a new algorithm for this type of problem, and described the development of a multi-purposes computer program. An example and the outputs of the multi-purposes computer program were also included.

The research results in this dissertation can be summarized as follows:

1) Five algorithms were developed in the Phase I research work. Seven separate computer programs were coded and tested. The best algorithm developed to solve the single-ship through single-prospect problem is the algorithm based upon the modification of Little's sequential tour-building method. The computer program for this algorithm takes only 60 K bytes to solve a 14-line configuration problem while the Willard algorithm takes 250 K

bytes to solve a 10-line configuration problem. The computation time for this algorithm is only slightly higher than Willard's.

2) An algorithm for solving the shortest Hamiltonian chain problem was developed in this dissertation, and proved to be better than an existing algorithm in the current literature. The algorithm developed can be applied to a general machine sequence scheduling problem with unknown starting and ending entries.

3) An algorithm was developed in the Phase II research work to solve both the single-ship through multi-prospects problem and the multi-ships through multi-prospects problem. A multi-purposes computer program was also coded which will provide a user several varieties of option.

Recommendations

There are three major areas for future research in the area of large scale marine seismic mapping operations.

The multi-purposes computer program developed in this dissertation was coded in the FORTRAN IV language. Various amounts of computer memory are required to use this computer program. Since many seismic ships are equipped with small memory digital computers, it is highly recommended that the computer program is re-coded into the Conversation Program System (CPS) language. The CPS language could be PL/1 language oriented, BASIC language oriented, or FORTRAN language oriented. The CPS language will enable a user to transmit, possibly through a remote communication facility, the input data from the terminal on the ship to a large memory computer on the mainland where the best route of a ship is determined. Then the result can be dispatched back to the ship. Another reason for re-coding the computer program into the CPS language is that the decision of routing a large scale marine seismic mapping operation might require the communication between the party manager aboard ship and the office executive manager. The CPS language can service this purpose.

The second area of research should be the modification of the procedure of calculating penalty matrix which is based upon the X and Y coordinates of each line to calculation based upon the longitude and the latitude of each line. Since the longitude and latitude are used for navigational purposes, it is suggested that a subroute which can convert the input data of longitude and latitude directly into penalty matrix shall be developed and included in the multi-purposes computer program in the future.

The third area of research should be directed toward developing algorithm that yields the optimal routing of the large scale ship mapping problem. As mentioned in Chapter VII, a heuristic algorithm has been developed in this dissertation for this problem. Since the sequence of prospect is predetermined before the routing process for each prospect begins, the algorithm may not reach an optimal solution. If all the prospects are apart from each other, an optimal solution is likely. However, if all the prospects are close to each other, a non-optimal (but near-optimal) solution is quite possible.

Although the primary result of this dissertation will be the reduction of the managerial decision making difficulties involved in scheduling geophysical ships, the result can also be extended to the solution of constrained traveling salesman problem and the machine sequence scheduling problem.

A SELECTED BIBLIOGRAPHY

(1) Arnoff, E. L., and S. S. Sengupta. "The Traveling Salesman Problem." Progress in Operations, Vol. I (1961), pp. 150-157.

(2) Baker, K. R. Introduction to Sequencing and Scheduling. New York: John Wiley and Sons, Inc., 1974.

(3) Balas, E. "An Additive Algorithm for Solving Linear Programs With Zero One Variables." Operations Research, Vol. 13 (1965), pp. 517-546.

(4) Barachet, L. L. "Graphic Solution of Traveling Salesman Problem." Operations Research, Vol. 5 (1957), pp. 841-845.

(5) Bellman, R. "Dynamic Programming Treatment of the Traveling Salesman Problem." Journal, Association for Computing Machinery, Vol. 9 (1962), pp. 61-63.

(6) Bellman, R. E. "On an Optimal Routing Problem." Quarterly Applied Mathematics, Vol. 16 (1958), pp. 87-90.

(7) Bellman, M., and G. L. Nemhauser. "The Traveling Salesman Problem: A Survey." Operations Research, Vol. 16 (1968), pp. 538-558.

(8) Bellmore, M., and J. C. Malone. "Pathology of Traveling-Salesman Subtour-Elimination Algorithms." Operations Research, Vol. 19 (1971), pp. 278-307.

(9) Bock, F. "Mathematical Programming Solution of Traveling Salesman Examples." Recent Advances in Mathematical Programming. New York: McGraw-Hill, 1963, pp. 65-75.

(10) Christofides, N. "The Shortest Hamiltonian Chain of a Graph." SIAM Journal of Applied Mathematics, Vol. 19, No. 4 (1970), pp. 689-696.

(11) Christofides, N. "Bound for the Traveling-Salesman Problem." Operations Research, Vol. 20, No. 5 (1972), pp. 1044-1056.

(12) Croes, G. A. "A Method for Solving Traveling Salesman Problem." Operations Research, Vol. 6 (1958), pp. 791-812.

(13) Dacey, M. F. "Selection of An Initial Solution for the Traveling Salesman Problem." Operations Research, Vol. 8 (1980), pp. 133-134.

(14) Dantzig, G. B., D. R. Fulkerson, and S. M. Johnson. "Solution of a Large Scale Traveling Salesman Problem." Operations Research, Vol. 2 (1954), pp. 393-410.

(15) Dantzig, G. B., D. R. Fulkerson, and S. M. Johnson. "On a Linear Programming Combinatorial Approach to the Traveling Salesman Problem." Operations Research, Vol. 7 (1959), pp. 58-66.

(16) Derman, C., and M. Klein. "Surveillance of Multicomponent System: A Stochastic Traveling Salesman's Problem." Naval Research Logistics Quarterly, Vol. 13 (1966), pp. 103-112.

(17) Dijkstra. "A Note on Two Problems in Connection with Graphs." Numerische Mathematik, Vol. 1 (1959), pp. 269-271.

(18) Dreyfus, S. E. "An Appraisal of Some Shortest-Path Algorithms." Operations Research, Vol. 17 (1969), pp. 395-412.

(19) Eastman, W. L. "Linear Programming With Pattern Constraints." (Unpublished Ph.D. Dissertation, Harvard University, 1958.)

(20) Eastman, W. L. "A Solution to the Traveling Salesman Problem." Presented at the American Summer Meeting of the Econometric Society, Cambridge, Massachusetts, August, 1958.

(21) Eilon, Samuel, S. D. T. Watson Gandy, and N. Christofides. Distribution Management: Mathematical Model and Practical Analysis. London: Griffin, 1971.

(22) Flood, M. M. "The Traveling Salesman Problem." Operations Research, Vol. 4 (1956), pp. 61-75.

(23) Garfinkel, R. "On Partitioning the Feasible Set in a Branch-and-Bound Algorithm for the Asymmetric Traveling Salesman Problem." Operations Research, Vol. 19 (1971), p. 340.

(24) Garfinkel, R., and G. L. Nemhauser. Integer Programming. New York: John Wiley and Sons, 1972.

(25) Gillett, B. E., and L. R. Miller. "A Heuristic Algorithm for the Vehicle Dispatch Problem." Operations Research, Vol. 22 (1974), pp. 340-349.

(26) Gilmore, P. C., and R. E. Gomory. "Sequencing a One-State Variable Machine: A Solvable Case of the Traveling Salesman Problem." Operations Research, Vol. 12 (1964), pp. 655-679.

(27) Gonzales, R. H. "Solution to the Traveling Salesman Problem by Dynamic Programming on the Hypercube." Technical Report No. 18. Cambridge, Massachusetts: Masachusetts Institute of Technology, Operations Research Center, 1962.

(28) Gorenstein, S. "Printing Press Scheduling for Multi-Edition Periodicals." Management Science, Vol. 16 (1970), B-373-383.

(29) Hansen, K. H., and J. Krarup. "Improvement of the Held-Karp Algorithm for the Symmetric Traveling-Salesman Problem." Mathematical Programming, Vol. 7 (1974), pp. 87-96.

(30) Hardgrave, W. W., and G. L. Nemhauser. "On the Relation Between the Traveling Salesman and the Longest Path Problem." Operations Research, Vol. 10 (1962), pp. 647-657.

(31) Held, M., and R. M. Karp. "A Dynamic Programming Approach to Sequencing Problem. SIAM Journal of Applied Mathematics, Vol. 10 (1962), pp. 196-210.

(32) Held, M., and R. M. Karp. "The Traveling-Salesman Problem and Minimum Spanning Trees, Part I." Operations Research, Vol. 18 (1970), pp. 1138-1162.

(33) Held, M., and R. M. Karp. "The Traveling-Salesman Problem and Minimum Spanning Trees, Part II." Mathematical Programs, Vol. 1 (1971), pp. 6-26.

(34) Hobson, G. D., and W. Pohl. Modern Petroleum Technology. New York: John Wiley and Sons, 1973.

(35) Hu, T. C. "A Decomposition Algorithm for Shortest Paths in a Network." Operations Research, Vol. 16 (1968), pp. 91-102.

(36) Isaac, A. M., and E. Turban. "Some Comments on the Traveling Salesman Problem." Operations Research, Vol. 17 (1969), pp. 543-546.

(37) Karg, L. L., and G. L. Thompson. "A Heuristic Approach to Solving Traveling Salesman Problem." Management Science, Vol. 10 (1964), pp. 225-248.

(38) Kruskal, J. B. "On the Shortest Spanning Subtree of a Graph and the Traveling Salesman Problem." Proceedings of the American Mathematical Society, Vol. 2 (1956), pp. 48-50.

(39) Lawler, E. L., and D. E. Wood. "Branch-and-Bound Methods: A Survey." Operations Research, Vol. 14 (1966), pp. 699-719.

(40) Lin, S. "Computer Solution of the Traveling Salesman Problem." Bell System Technical Journal, Vol. 44 (1965), pp. 2245-2269.

(41) Lin, S., and B. W. Kerninghan. "An Effective Heuristic Algorithm for the Traveling Salesman Problem." Operations Research, Vol. 20 (1972), pp. 498-516.

(42) Little, J. D. C., K. G. Murty, D. W. Sweeney, and C. Karel. "An Algorithm for the Traveling Salesman Problem." Operations Research, Vol. 11 (1963), pp. 979-989.

(43) Martin, G. T. "Solving the Traveling Salesman Problem by Integer Linear Programming. Journal of Operations Research, Sec. 14 (1966), B-171.

(44) Miller, C. E., A. W. Tucker, and R. A. Zemlin. "Integer Programming Formulations and Traveling Salesman Problem." Association of Computing Machinery Journal, Vol. 7 (1960), pp. 326-329.

(45) Netter, J. P. "An Algorithm to Find Elementary Negative-Cost Circuits With a Given Number of Arcs-The Traveling-Salesman Problem." Operations Research, Vol. 19, Pt. 1 (1971), pp. 234-237.

(46) Nicholson, T. A. "Finding the Shortest Route Between Two Points in a Network." Computer Journal, Vol. 9 (1966), pp. 275-280.

(47) Obruca, A. K. "Spanning Tree Manipulation and the Traveling Salesman Problem." Computer Journal, Vol. 10 (1968), pp. 374-377.

(48) Peart, R. M., R. M. Randolph, and T. E. Bartlett. "The Shortest Route Problem." Operations Research, Vol. 8 (1960), pp. 866-868.

(49) Perko, A. "Some Computational Notes on the Shortest Route Problem." Computer Journal, Vol. 8 (1965), p. 19.

(50) Pollack, M., and W. Weibenson. "Solution of the Shortest Route Problem - A Survey." Operations Research, Vol. 8 (1960), pp. 224-230.

(51) Prime, R. C. "Shortest Connection Networks and Some Generalization." Bell System Technical Journal, Vol. 36 (1957), pp. 1389-1401.

(52) Reiter, S., and G. Sherman. "Discrete Optimizing." Operations Research, Vol. 11 (1963), pp. 979-989.

(53) Roberts, S. M., and B. Flores. "An Engineering Approach to the Traveling Salesman Problem." Management Science, Vol. 13 (1966), pp. 269-288.

(54) Roger, J. H., and R. G. Carpenter. "The Cumulative Construction of Minimum Spanning Trees." Applied Statistics (1974), pp. 192-206.

(55) Rothkopf, M. "The Traveling Salesman Problem: On the Reduction of Certain Large Problem to Smaller Ones." Operations Research, Vol. 14 (1966), pp. 532-533.

(56) Rossman, M. J., R. J. Twery, and F. D. Stone. "A Solution to the Traveling Salesman Problem by Combinational Programming." Operations Research, Vol. 6 (1958), pp. 897.

(57) Saksena, J. P., and S. Human. "Routing Problem With K Specified Nodes." Operations Research, Vol. 14 (1966), pp. 909-913.

(58) Seppanen, J. J. "Spanning Tree." Communications Assocation of Computing Machinery, Algorithm 399 (1970), 1 p.

(59) Shapiro, D. "Algorithms for the Solution of the Optimal Cost Traveling Salesman Problem." (Unpublished Sc.D. Thesis, Washington University, 1966.)

(60) Steckhan, H. "A Theorem on Symmetric Traveling Salesman Problems." Operations Research, Vol. 18, Pt. 2 (1970), pp. 1163-1167.

(61) Svestka, J. A., and V. E. Huckfeldt. "Computer Experience With An M-Salesman Traveling Salesman Algorithm." Management Science, Vol. 19, No. 7 (1973), pp. 790-799.

(62) Tillman, F., and H. Chochran. "A Heuristic Approach for Solving the Delivery Problem." Journal of Industrial Engineering, Vol. 19 (1968), pp. 354.

(63) Turner, W. C., and P. M. Ghare, and L. R. Fourds. "Transportation Routing Problem - A Survey." AIIE Transactions, Vol. 6, No. 4 (December, 1974), pp. 288-301.

(64) Turner, W. C. "Vehicle Routing Classification, Characteristics and Solution Procedures." (An unpublished working paper, Oklahoma State University, 1975.)

(65) Wagner, H. M. Principles of Operations Research With Application. Englewood Cliffs, New Jersey: Prentice-Hall, 1969.

(66) Willard, E. P. "An Algorithm for Optimal Ship Routing for Seismic Data Collection." (Unpublished Ph.D. Dissertation, Oklahoma State University, 1970.)

(67) Williams, M. L. "Solution of the Multiple-Constraint Allocation Problem Using Recursive Search Dynamic Programming." (Unpublished Ph.D. Dissertation, Oklahoma State University, 1971.)

APPENDIX

SOURCE LISTING OF MULTI-PURPOSES COMPUTER PROGRAM FOR LARGE SCALE MARINE SEISMIC MAPPING OPERATIONS

```
C
C     MULTI-PURPOSE COMPUTER PROGRAM FOR SHIP ROUTING PROBLEM
C
      IMPLICIT INTEGER (A-V,Z)
      REAL SORT,TM,SR,CABLE,SPIN,SPOUT,CABOUT,CABIN,SCALE
      COMMON /M1/B(3,15,2,15,2)/M2/IINF(2000),IPROB(2000)
      COMMON /V1/IIKEY(15),JJKEY(15),KKKEY(15),LLKEY(15)
      COMMON /S1/M,MM,N,R,S,BCUNC/S2/MAX,ICOST,KCOST,LK,TB,GCOST,BCOST
      COMMON /S3/TM,SR
      COMMON /R1/WX(5,15,2),WY(5,15,2)/R2/WXP(2),WYP(2)/R3/XC(5,2),
     1       YC(5,2)/R4/XXP(5,2),YYP(5,2)/R5/NL(5),PATH(5)
      COMMON /R6/MAP,CABLE,SPIN,SPOUT,CABOUT,CABIN,SCALE,IP,IS,IPP
      COMMCN /R9/JOVER(5),OL1(5,7),OE1(5,7),OL2(5,7),OE2(5,7),XDIST(5,7)
      IPP=0
      READ(5,290)M,SCALE,TM,SR
  290 FORMAT(I10,3F10.3)
      READ(5,300)IS,OPT
  300 FORMAT(2I10)
      DO 1000 IG=1,IS
      READ(5,304)MAP,IP
  304 FORMAT(2I10)
      IPP=IPP+IP
      READ(5,305)CABLE,SPIN,SPOUT,CABCUT,CABIN
  305 FORMAT(5F10.3)
      READ(5,310)WXP(1),WYP(1),WXP(2),WYP(2)
  310 FORMAT(4F10.3)
      DO 320 I=1,IP
      K=1
      NL(I)=1
  315 READ(5,325)WX(I,K,1),WY(I,K,1),WX(I,K,2),WY(I,K,2),KBTA
  325 FORMAT(4F10.3,I10)
      IF(KBTA.EQ.99)GO TO 321
      NL(I)=NL(I)+1
      K=K+1
      GO TO 315
  321 J=1
   46 READ(5,40)JBTA,OL1(I,J),OE1(I,J),OL2(I,J),OE2(I,J),XDIST(I,J)
   40 FORMAT(5I10,F10.3)
      IF(JBTA.EQ.99)GO TO 45
      J=J+1
      GO TO 46
   45 JOVER(I)=J-1
  320 CONTINUE
      CALL OTPUT1(IG)
      IF(OPT.EQ.2)GO TO 3000
      XXP(1,1)=WXP(1)
      YYP(1,1)=WYP(1)
      IF(IP.EQ.1)GO TO 550
      CALL SPATH
      DO 500 KA=1,IP
      KAA=PATH(KA)
      IF(KA.EQ.IP)GO TO 544
      LINE1=NL(KAA)+1
      KA1=KA+1
      KG=PATH(KA1)
      XXP(KA,2)=XC(KG,1)
      YYP(KA,2)=YC(KG,1)
      GO TO 545
```

```
 544 XXP(KA,2)=WXP(2)
     YYP(KA,2)=WYP(2)
 545 CALL MATRIX(KAA,KA)
     CALL ROUTE(KAA,1)
     CALL OTPUT2(KAA,KA,1)
     IF(KA.EQ.IP)GO TO 500
     I1=IIKEY(LINE1)-1
     XXP(KA1,1)=WX(KAA,I1,JJKEY(LINE1))
     YYP(KA1,1)=WY(KAA,I1,JJKEY(LINE1))
 500 CONTINUE
     GO TO 1000
 550 XXP(1,2)=WXP(2)
     YYP(1,2)=WYP(2)
     CALL MATRIX(1,1)
     CALL ROUTE(1,1)
     CALL OTPUT2(1,1,1)
     GO TO 1000
3000 READ(5,3501)(PATH(K),K=1,IP)
3501 FORMAT(10I10)
     DO 2000 I=1,IP
     READ(5,2001)XXP(I,1),YYP(I,1),XXP(I,2),YYP(I,2)
2001 FORMAT(4F10.3)
     KAA=PATH(I)
     LINES=NL(KAA)+1
     DO 2500 J=1,LINES
     READ(5,2002)IIKEY(J),JJKEY(J),KKKEY(J),LLKEY(J)
2002 FORMAT(4I5)
2500 CONTINUE
     CALL MATRIX(KAA,I)
     CALL OTPUT2(KAA,I,2)
2000 CONTINUE
1000 CONTINUE
     STOP
     END
```

```
      SUBROUTINE OTPUT1(KG)
      IMPLICIT INTEGER (A-V,Z)
      REAL SQRT,CABLE,SPIN,SPOUT,CABOUT,CABIN,SCALE
      DIMENSION ZL(20)
      COMMON /R1/WX(5,15,2),WY(5,15,2)/R2/WXP(2),WYP(2)/R3/XC(5,2),
     1        YC(5,2)/R4/XXP(5,2),YYP(5,2)/R5/NL(5),PATH(5)
      COMMON /R6/MAP,CABLE,SPIN,SPOUT,CABOUT,CABIN,SCALE,IP,IS,IPP
      DATA ZL/1HA,1HB,1HC,1HD,1HE,1HF,1HG,1HH,1HI,1HJ,1HK,1HL,1HM,1HN,
     1         1HO,1HP,1HQ,1HR,1HS,1HT/
C
C  ***********************************************************************
C  PRINT ROUTINE  -  COORDINATES AND PARAMETERS
C  ***********************************************************************
C
      WRITE(6,99)
   99 FORMAT(1H1,////)
      WRITE(6,100)
  100 FORMAT(1H ,14X,48H************************************************
     1)
      WRITE(6,101)
      WRITE(6,101)
  101 FORMAT(1H ,14X,1H*,46X,1H*)
      WRITE(6,102)IS,IPP
  102 FORMAT(1H ,14X,1H*,I3,14H-SHIPS THROUGH,I2,26H-PROSPECTS ROUTING P
     1ROBLEM,2H *)
      WRITE(6,101)
      WRITE(6,101)
      WRITE(6,100)
      WRITE(6,303)KG
  303 FORMAT(1H ,////,34X,8HSHIP NO:,I3///)
      IF(MAP.EQ.1)GO TO 754
      WRITE(6,360)
  360 FORMAT(15X,48H************************************************)
      WRITE(6,361)
  361 FORMAT( 15X,1H*,46X,1H*)
      WRITE(6,361)
      WRITE(6,362)
  362 FORMAT(15X,1H*,16X,15HSHIP PARAMETERS,15X,1H*)
      WRITE (6,361)
      WRITE (6,361)
      KCABLE = CABLE
      WRITE (6,363) KCABLE
  363 FORMAT(15X, 1H*, 11X, 14HCABLE LENGTH = , I5, 5H FEET, 11X, 1H*)
      WRITE (6,364)  SCALE
  364 FORMAT(15X, 1H*, 11X, 8HSCALE = , F5.2, 11H UNITS/MILE, 11X, 1H*)
      WRITE (6,361)
      WRITE (6,365)  SPOUT
  365 FORMAT(15X, 1H*, 6X, 25HSHIP SPEED (CABLE OUT) = , F5.2, 4H MPH,
     1 6X, 1H*)
      WRITE (6,366)  SPIN
  366 FORMAT(15X, 1H*, 6X, 25HSHIP SPEED (CABLE IN)  = , F5.2, 4H MPH,
     1 6X, 1H*)
      WRITE (6,361)
      WRITE (6,367)  CABOUT
  367 FORMAT(15X, 1H*, 2X, 32HCABLE HANDLING TIME (LAY OUT) = , F4.2,
     1 9H HOURS   *)
      WRITE (6,368)  CABIN
  368 FORMAT(15X, 1H*, 2X, 32HCABLE HANDLING TIME (PULL IN) = , F4.2,
     1 9H HOURS  *)
      WRITE (6,361)
      WRITE (6,361)
  754 WRITE (6,360)
      WRITE(6,361)
      WRITE(6,361)
      WRITE(6,750)
  750 FORMAT(15X,1H*,15X,17HPORT CO-ORDINATES,14X,1H*)
      WRITE(6,361)
      WRITE(6,361)
      WRITE (6,755)
  755 FORMAT(15X, 1H*, 16X, 2HX1, 4X, 2HY1, 8X, 2HX2, 4X, 2HY2, 6X, 1H*)
      WRITE (6,760)
  760 FORMAT (1H+, 30X, 10H____   ____, 6X, 10H____   ____)
      WRITE(6,361)
      WRITE(6,767)WXP(1),WYP(1),WXP(2),WYP(2)
  767 FORMAT(15X, 1H*, 5X, 1HP,  9X, F4.0, 2X, F4.0, 6X, F4.0, 2X, F4.0,
     1 5X, 1H*)
      WRITE(6,361)
      WRITE(6,361)
      WRITE(6,360)
      DO 50 I=1,IP
      WRITE(6,361)
      WRITE(6,361)
      WRITE(6,299)I
  299 FORMAT(15X,1H*,16X,12HPROSPECT NO:,I3,15X,1H*)
      WRITE(6,361)
      WRITE(6,361)
      WRITE(6,755)
      WRITE(6,760)
      WRITE(6,361)
      LINE1=NL(I)
      DO 51 K=1,LINE1
      WRITE(6,765)ZL(K),WX(I,K,1),WY(I,K,1),WX(I,K,2),WY(I,K,2)
  765 FORMAT(15X,1H*,5X,A1,9X,F4.0,2X,F4.0,6X,F4.0,2X,F4.0,5X,1H*)
   51 CONTINUE
      WRITE(6,361)
      WRITE(6,361)
      WRITE(6,360)
   50 CONTINUE
C
C     SCALE THE CO-ORDINATES
C
   55 DO 60 J=1,2
      WXP(J)=WXP(J)/SCALE
      WYP(J)=WYP(J)/SCALE
   60 CONTINUE
      DO 70 I=1,IP
      LINE2=NL(I)
      DO 70 J=1,LINE2
      DO 70 K=1,2
      WX(I,J,K)=WX(I,J,K)/SCALE
      WY(I,J,K)=WY(I,J,K)/SCALE
   70 CONTINUE
      RETURN
      END
      SUBROUTINE SPATH
      IMPLICIT INTEGER (A-V,Z)
      REAL SQRT
      COMMON /M1/B(3,15,2,15,2)/M2/IINF(2000),IPROB(2000)
      COMMON /V1/IIKEY(15),JJKEY(15),KKKEY(15),LLKEY(15)
      COMMON /S1/M,MM,N,R,S,BOUND/S2/MAX,ICOST,KCOST,LK,TB,GCOST,BCOST
      COMMON /R1/WX(5,15,2),WY(5,15,2)/R2/WXP(2),WYP(2)/R3/XC(5,2),
     1       YC(5,2)/R4/XXP(5,2),YYP(5,2)/R5/NL(5),PATH(5)
      COMMON /R6/MAP,CABLE,SPIN,SPOUT,CABOUT,CABIN,SCALE,IP,IS,IPP
      IP1=IP+1
      DO 10 I=1,IP
      XT=0.0
      YT=0.0
      LINE1=NL(I)
      DO 15 J=1,LINE1
      XT=XT+WX(I,J,1)+WX(I,J,2)
      YT=YT+WY(I,J,1)+WY(I,J,2)
   15 CONTINUE
      ZNL=2*NL(I)
      XC(I,1)=(XT/ZNL)
      XC(I,2)=XC(I,1)+1.
      YC(I,1)=(YT/ZNL)
      YC(I,2)=YC(I,1)+1.
   10 CONTINUE
      DO 51 K=1,IP
   51 WRITE(6,100)XC(K,1),YC(K,1)
  100 FORMAT(1H0,'XC-YC',2F10.3)
      B(1,1,1,1,1)=M
      B(1,1,2,1,2)=M
      B(1,1,1,1,2)=SQRT((WXP(1)-WXP(2))**2+(WYP(1)-WYP(2))**2)+0.5
      B(1,1,2,1,1)=B(1,1,1,1,2)
      DO 20 I=1,IP
      I1=I+1
      DO 20 J=1,2
      DO 20 K=1,IP
      K1=K+1
      DO 20 L=1,2
      IF((I.EQ.K).AND.(J.EQ.L)) GO TO 25
      B(1,I1,J,K1,L)=SQRT(((XC(I,J)-XC(K,L))**2+((YC(I,J)-YC(K,L))**2))
     1+0.5
      GO TO 20
   25 B(1,I1,J,K1,L)=M
   20 CONTINUE
      DO 35 I=1,IP
      DO 35 J=1,2
      I1=I+1
      B(1,1,1,I1,J)=SQRT(((WXP(1)-XC(I,J))**2)+((WYP(1)-YC(I,J))**2))
     1+0.5
```

```
      B(1,I1,J,1,1)=B(1,1,1,I1,J)
      B(1,1,2,I1,J)=SQRT(((WXP(2)-XC(I,J))**2)+((WYP(2)-YC(I,J))**2))
     1+0.5
      B(1,I1,J,1,2)=B(1,1,2,I1,J)
   35 CONTINUE
      DO 300 I=1,IP1
      DO 300 J=1,2
  300 WRITE(6,400)((B(1,I,J,K,L),L=1,2),K=1,IP1)
  400 FORMAT(1H ,20I6)
      CALL ROUTE(IP1,2)
      DO 50 I=1,IP
      PATH(I)=KKKEY(I)-1
      WRITE(6,200)I,PATH(I)
  200 FORMAT(1H0,'I-PATH',2I10)
   50 CONTINUE
      RETURN
      END
      SUBROUTINE MATRIX(KAA,KA)
      IMPLICIT INTEGER (A-V,Z)
      REAL SQRT,CHTIME,BEDIST,SIDE1,SIDE2,TAN,CTN,DLINE,TIME,
     1     CABLE,SPIN,SPOUT,CABOUT,CABIN,SCALE
      DIMENSION KL(15)
      COMMON /M1/B(3,15,2,15,2)/M2/IINF(2000),IPROB(2000)
      COMMON /S1/M,MM,N,R,S,BOUND/S2/MAX,ICOST,KCOST,LK,TB,GCOST,BCOST
      COMMON /R1/WX(5,15,2),WY(5,15,2)/R2/WXP(2),WYP(2)/R3/XC(5,2),
     1     YC(5,2)/R4/XXP(5,2),YYP(5,2)/R5/NL(5),PATH(5)
      COMMON /R6/MAP,CABLE,SPIN,SPOUT,CABOUT,CABIN,SCALE,IP,IS,IPP
      COMMON /R7/X(15,2),Y(15,2),XX(15,2),YY(15,2)/R8/TAN(15),CTN(15),
     1       DLINE(15),TIME(15),BEDIST
      COMMON /R9/JOVER(5),OL1(5,7),OE1(5,7),OL2(5,7),OE2(5,7),XDIST(5,7)
      DATA KL/1HP,1HA,1HB,1HC,1HD,1HE,1HF,1HG,1HH,1HI,1HJ,1HK,1HL,1HM,
     1       1HN/
C
C   *******************************************************************
C   CALCULATION OF NEW CO-ORDINATES
C   *******************************************************************
      LINES=NL(KAA)
      LINE1=LINES+1
      DO 21 I=1,LINES
      DO 21 J=1,2
      X(I,J)=WX(KAA,I,J)
      Y(I,J)=WY(KAA,I,J)
   21 CONTINUE
      XP1=XXP(KA,1)
      YP1=YYP(KA,1)
      XP2=XXP(KA,2)
      YP2=YYP(KA,2)
      CHTIME = CABOUT + CABIN
      BEDIST = CHTIME/((1./SPOUT)-(1./SPIN))
      CABLE = (CABLE/5280.)*2.
      IF(MAP.EQ.1)  GO TO 66
      DO 60  K=1,LINES
      IF(X(K,2).EQ.X(K,1))  GO TO 61
      IF(Y(K,2).EQ.Y(K,1))  GO TO 62
      TAN(K) = (Y(K,2)-Y(K,1))/(X(K,2)-X(K,1))
      SIDE1 = SQRT((CABLE**2.)/(1.+(TAN(K)**2 )))
      XX(K,1) = X(K,1) + SIDE1
      XX(K,2) = X(K,2) - SIDE1
      IF(X(K,1).GT.X(K,2))  GO TO 60
      XX(K,1) = X(K,1) - SIDE1
      XX(K,2) = X(K,2) + SIDE1
      GO TO 60
   61 XX(K,1) = X(K,1)
      XX(K,2) = X(K,2)
      YY(K,1) = Y(K,1) - CABLE
      YY(K,2) = Y(K,2) + CABLE
      IF(Y(K,2).GT.Y(K,1))  GO TO 60
      YY(K,1) = Y(K,1) + CABLE
      YY(K,2) = Y(K,2) - CABLE
      GO TO 60
   62 YY(K,1) = Y(K,1)
```

```
      YY(K,2) = Y(K,2)
      XX(K,1) = X(K,1) - CABLE
      XX(K,2) = X(K,2) + CABLE
      IF(X(K,2).GT.X(K,1))  GO TO 60
      XX(K,1) = X(K,1) + CABLE
      XX(K,2) = X(K,2) - CABLE
      GO TO 60
   60 CONTINUE
      DO 65  K=1,LINES
      IF(X(K,2).EQ.X(K,1))  GO TO 65
      IF(Y(K,2).EQ.Y(K,1))  GO TO 65
      CTN(K) = 1./TAN(K)
      SIDE2 = SQRT((CABLE**2.)/(1.+(CTN(K)**2 )))
      YY(K,1) = Y(K,1) + SIDE2
      YY(K,2) = Y(K,2) - SIDE2
      IF(Y(K,1).GT.Y(K,2))  GO TO 65
      YY(K,1) = Y(K,1) - SIDE2
      YY(K,2) = Y(K,2) + SIDE2
   65 CONTINUE
C
C
C
C     ********************************************************************
C     CALCULATICN OF PENALTY MATRIX
C     ********************************************************************
   66 B(1,1,1,1,1)=M
      B(1,1,2,1,2)=M
      B(1,1,2,1,1)=SQRT((XP1-XP2)**2+(YP1-YP2)**2)+0.5
      B(1,1,1,1,2)=B(1,1,2,1,1)
      IF(MAP.EQ.0)GO TO 70
      DO 42 I=1,LINES
      I1=I+1
      DO 42 J=1,2
      DO 42 K=1,LINES
      K1=K+1
      DO 42 L=1,2
      IF((I.EQ.K).AND.(J.EQ.L))GO TO 41
      B(1,I1,J,K1,L)=SQRT(((X(I,J)-X(K,L))**2)+((Y(I,J)-Y(K,L))**2))+0.5
      GO TO 42
   41 B(1,I1,J,K1,L)=M
   42 CONTINUE
      DO 45 I=1,LINES
      DO 45 J=1,2
      I1=I+1
      B(1,1,1,I1,J)=SQRT(((XP1-X(I,J))**2)+((YP1-Y(I,J))**2))+0.5
      B(1,I1,J,1,1)=B(1,1,1,I1,J)
      B(1,1,2,I1,J)=SQRT(((XP2-X(I,J))**2)+((YP2-Y(I,J))**2))+0.5
      B(1,I1,J,1,2)=B(1,1,2,I1,J)
   45 CONTINUE
      GO TO 90
   70 DO 52 I=1,LINES
      I1=I+1
      DO 52 J=1,2
      DO 52 K=1,LINES
      K1=K+1
      DO 52 L=1,2
      IF((I.EQ.K).AND.(J.EQ.L))GO TO 51
      B(1,I1,J,K1,L)=SQRT(((X(I,J)-XX(K,L))**2)+((Y(I,J)-YY(K,L))**2))
   1+0.5
    IF(I.NE.K)GO TO 52
    DLINE(I)=SQRT(((X(I,1)-X(I,2))**2)+((Y(I,1)-Y(I,2))**2))
    TIME(I)=DLINE(I)/SPOUT
    GO TO 52
 51 B(1,I1,J,K1,L)=M
 52 CONTINUE
    DO 55 I=1,LINES
    DO 55 J=1,2
    I1=I+1
    B(1,1,1,I1,J)=SQRT(((XP1-XX(I,J))**2)+((YP1-YY(I,J))**2))+0.5
    B(1,I1,J,1,1)=B(1,1,1,I1,J)
    B(1,1,2,I1,J)=SQRT(((XP2-X(I,J))**2)+((YP2-Y(I,J))**2))+0.5
    B(1,I1,J,1,2)=B(1,1,2,I1,J)
 55 CONTINUE
 90 IF(JOVER(KAA).EQ.0)GO TO 33
    IN=JOVER(KAA)
    DO 67 I=1,IN
    KL1=OL1(KAA,IN)+1
    KE1=OE1(KAA,IN)
    KL2=OL2(KAA,IN)+1
    KE2=OE2(KAA,IN)
    B(1,KL1,KE1,KL2,KE2)=XDIST(KAA,IN)/SCALE
 67 CONTINUE
 33 IF(MAP.EQ.1)GO TO 32
    DO 85 I=1,LINE1
    DO 85 J=1,2
    WB=B(1,1,1,I,J)
    B(3,1,1,I,J)=WB+CABLE
    B(3,I,J,1,1)=B(3,1,1,I,J)
    B(3,1,2,I,J)=B(1,1,2,I,J)
    B(3,I,J,1,2)=B(3,1,2,I,J)
    WB=B(1,1,1,I,J)
    B(1,1,1,I,J)=WB/SPIN+CABOUT
    B(1,I,J,1,1)=B(1,1,1,I,J)
    WB=B(1,1,2,I,J)
    B(1,1,2,I,J)=WB/SPIN+CABIN
    B(1,I,J,1,2)=B(1,1,2,I,J)
 85 CONTINUE
    DO 86 I=2,LINE1
    DO 86 J=1,2
    DO 86 K=2,LINE1
    DO 86 L=1,2
    WB=B(1,I,J,K,L)
    B(3,I,J,K,L)=WB+CABLE
    IF(I.EQ.K)GO TO 88
    IF(WB.LE.BEDIST) GO TO 87
    B(1,I,J,K,L)=WB/SPIN+CHTIME
    GO TO 86
 87 B(1,I,J,K,L)=(WB+CABLE)/SPOUT
    GO TO 86
 88 IF(J.EQ.L)GO TO 86
    B(1,I,J,K,L)=WB/SPOUT
 86 CONTINUE
C
C
C
C   *********************************************************************
```

```
C  PRINT ROUTINE  -  PENALTY MATRIX
C  *********************************************************************
   32 LINE2=LINES-2
      GO TO (3,4,5,6,7,8,9,10,11,12,13,14),LINE2
C
    3 WRITE (6,72)
   72 FORMAT(1H1,1X,////)
      WRITE (6,3331)  KAA
 3331 FORMAT ( 18X, 32H PENALTY MATRIX FOR PROSPECT NO:,I2///)
      WRITE (6,3332)
 3332 FORMAT (13X,  47H      P2    A1    A2    B1    B2    C1    C2    )
      WRITE (6,3333)
 3333 FORMAT ( 10X, 109H     *******************************************
     1*                                                             )
      GO TO 8000
C
    4 WRITE (6,72)
      WRITE (6,3341)  KAA
 3341 FORMAT (24X, 32H PENALTY MATRIX FOR PROSPECT NO:,I2///)
      WRITE (6,3342)
 3342 FORMAT ( 13X,  59H     P2    A1    A2    B1    B2    C1    C2    D
     11    D2    )
      WRITE (6,3343)
 3343 FORMAT ( 10X, 109H     *******************************************
     1*************                                                 )
      GO TO 8000
C
    5 WRITE (6,72)
      WRITE (6,3351)  KAA
 3351 FORMAT (30X, 32H PENALTY MATRIX FOR PROSPECT NO:,I2///)
      WRITE (6,3352)
 3352 FORMAT ( 13X,  71H     P2    A1    A2    B1    B2    C1    C2    D
     11    D2    E1    E2    )
      WRITE (6,3353)
 3353 FORMAT ( 10X, 109H     *******************************************
     1*************************                                     )
      GO TO 8000
C
    6 WRITE (6,72)
      WRITE (6,3361)  KAA
 3361 FORMAT ( 36X, 32H PENALTY MATRIX FOR PROSPECT NO:,I2///)
      WRITE (6,3362)
 3362 FORMAT (13X,  83H      P2    A1    A2    B1    B2    C1    C2    D
     11    D2    E1    E2    F1    F2    )
      WRITE (6,3363)
 3363 FORMAT (10X, 109H      *******************************************
     1*************************************                         )
      GO TO 8000
C
    7 WRITE (6,72)
      WRITE (6,3371)  KAA
 3371 FORMAT (39X, 32H PENALTY MATRIX FOR PROSPECT NO:,I2///)
      WRITE (6,3372)
 3372 FORMAT (13X,  95H      P2    A1    A2    B1    B2    C1    C2    D
     11    D2    E1    E2    F1    F2    G1    G2    )
      WRITE (6,3373)
 3373 FORMAT (10X, 109H      *******************************************
     1*************************************************             )
```

```
      GO TO 8000
C
    8 WRITE (6,72)
      WRITE (6,3381)  KAA
 3381 FORMAT (45X, 32H PENALTY MATRIX FOR PROSPECT NO:,I2///)
      WRITE (6,3382)
 3382 FORMAT (13X, 107H      P2     A1     A2     B1     B2     C1     C2     D
     11     D2     E1     E2     F1     F2     G1     G2     H1     H2    )
      WRITE (6,3383)
 3383 FORMAT (10X, 111H      ******************************************
     1************************************************************** )
      GO TO 8000
C
    9 WRITE(6,72)
      WRITE(6,3391)  KAA
 3391 FORMAT(23X,32H PENALTY MATRIX FOR PROSPECT NO:,I2///)
      WRITE(6,3392)
 3392 FORMAT(3X, 83H     P2  A1  A2  B1  B2  C1  C2  D1  D2  E1  E2  F1
     1  F2  G1  G2  H1  H2  I1  I2   )
      WRITE(6,3393)
 3393 FORMAT (1H ,  86H      ******************************************
     1**************************************)
      GO TO 8000
C
   10 WRITE(6,72)
      WRITE(6,3401)  KAA
 3401 FORMAT(27X,32H PENALTY MATRIX FOR PROSPECT NO:,I2///)
      WRITE(6,3402)
 3402 FORMAT(3X, 91H     P2  A1  A2  B1  B2  C1  C2  D1  D2  E1  E2  F1
     1  F2  G1  G2  H1  H2  I1  I2  J1  J2   )
      WRITE(6,3403)
 3403 FORMAT (1H ,  94H      ******************************************
     1**********************************************)
      GO TO 8000
C
   11 WRITE(6,72)
      WRITE(6,3411)  KAA
 3411 FORMAT(31X,32H PENALTY MATRIX FOR PROSPECT NO:,I2///)
      WRITE(6,3412)
 3412 FORMAT(3X, 99H     P2  A1  A2  B1  B2  C1  C2  D1  D2  E1  E2  F1
     1  F2  G1  G2  H1  H2  I1  I2  J1  J2  K1  K2   )
      WRITE(6,3413)
 3413 FORMAT (1H , 102H      ******************************************
     1******************************************************)
      GO TO 8000
C
   12 WRITE(6,72)
      WRITE(6,3421)  KAA
 3421 FORMAT(35X,32H PENALTY MATRIX FOR PROSPECT NO:,I2///)
      WRITE(6,3422)
 3422 FORMAT(3X,107H     P2  A1  A2  B1  B2  C1  C2  D1  D2  E1  E2  F1
     1  F2  G1  G2  H1  H2  I1  I2  J1  J2  K1  K2  L1  L2   )
      WRITE(6,3423)
 3423 FORMAT (1H , 110H      ******************************************
     1**************************************************************)
      GO TO 8000
C
   13 WRITE(6,72)
```

```
      WRITE(6,3431)  KAA
 3431 FORMAT(39X,32H PENALTY MATRIX FOR PROSPECT NO:,I2///)
      WRITE(6,3432)
 3432 FORMAT(3X,115H      P2   A1   A2   B1   B2   C1   C2   D1   D2   E1   E2   F1
     1  F2   G1   G2   H1   H2   I1   I2   J1   J2   K1   K2   L1   L2   M1   M2    )
      WRITE(6,3433)
 3433 FORMAT (1H , 118H      *******************************************
     1******************************************************************
     2***)
      GO TO 8000
C
   14 WRITE(6,72)
      WRITE(6,3441)  KAA
 3441 FORMAT(43X,32H PENALTY MATRIX FCR PROSPECT NO:,I2///)
      WRITE(6,3442)
 3442 FORMAT(3X,123H      P2   A1   A2   B1   B2   C1   C2   D1   D2   E1   E2   F1
     1  F2   G1   G2   H1   H2   I1   I2   J1   J2   K1   K2   L1   L2   M1   M2   N1
     2N2    )
      WRITE(6,3443)
 3443 FORMAT ( 1H , 126H     *******************************************
     1******************************************************************
     2***********)
 8000 DO 47 I=1,LINE1
      DO 47 J=1,2
      IF((I.EQ.1).AND.(J.EQ.2))GO TO 47
      IF(LINES.GE.9)GO TO 77
      WRITE(6,49)KL(I),J
   49 FORMAT(1H0,12X,A1,I1,1X,1H*)
      WRITE(6,91)B(1,I,J,1,2)
   91 FORMAT(1H+,16X,I4,2X)
      WRITE(6,46)((B(1,I,J,K,L),L=1,2),K=2,LINE1)
   46 FORMAT(1H+,22X,20(I4,2X))
      GO TO 47
   77 WRITE(6,79)KL(I),J
   79 FORMAT(1H0, 2X,A1,I1,1X,1H*)
      WRITE(6,71)B(1,I,J,1,2)
   71 FORMAT(1H+, 6X,I3,1X)
      WRITE(6,78)((B(1,I,J,K,L),L=1,2),K=2,LINE1)
   78 FORMAT(1H+,10X,20(I3,1X))
   47 CONTINUE
      IF(IP.EQ.1)RETURN
      IF(KA.EQ.1)GO TO 148
      IF(KA.EQ.IP)GO TO 149
      WRITE(6,180)
  180 FORMAT(1H ,////,13X,'NCTE: P1 IS AN ARTIFICAL STARTING PORT AND P2
     1 IS AN ARTIFICAL ENDING PORT.')
      RETURN
  148 WRITE(6,160)
  160 FORMAT(1H ,////,13X,'NOTE: P2 IS AN ARTIFICAL ENDING PORT.')
      RETURN
  149 WRITE(6,170)
  170 FORMAT(1H ,////,13X,'NOTE: P1 IS AN ARTIFICAL STARTING PORT.')
      RETURN
      END
      SUBROUTINE OTPUT2(KAA,KA,OPTION)
      IMPLICIT INTEGER (A-V,Z)
      REAL SQRT,CHTIME,BEDIST,SIDE1,SIDE2,TAN,CTN,DLINE,TIME,
     1     CABLE,SPIN,SPOUT,CABOUT,CABIN,SCALE
      DIMENSION KL(15),CABPOS(2),NB(5)
      COMMON /M1/B(3,15,2,15,2)/M2/IINF(2000),IPROB(2000)
      COMMON /V1/IIKEY(15),JJKEY(15),KKKEY(15),LLKEY(15)
      COMMON /S1/M,MM,N,R,S,BOUND/S2/MAX,ICOST,KCOST,LK,TB,GCOST,BCOST
      COMMON /R1/WX(5,15,2),WY(5,15,2)/R2/WXP(2),WYP(2)/R3/XC(5,2),
     1      YC(5,2)/R4/XXP(5,2),YYP(5,2)/R5/NL(5),PATH(5)
      COMMON /R6/MAP,CABLE,SPIN,SPOUT,CABOUT,CABIN,SCALE,IP,IS,IPP
      COMMON /R7/X(15,2),Y(15,2),XX(15,2),YY(15,2)/R8/TAN(15),CTN(15),
     1       DLINE(15),TIME(15),BEDIST
      DATA CABPOS/3HOUT,3H IN/
      DATA KL/1HP,1HA,1HB,1HC,1HD,1HE,1HF,1HG,1HH,1HI,1HJ,1HK,1HL,1HM,
     1      1HN/
      DATA NB/2H#1,2H#2,2H#3,2H#4,2H#5/
C
C
C  *********************************************************************
C  PRINT ROUTINE  -  OPTIMUM PATH
C  *********************************************************************
      LINES=NL(KAA)
      LINE1=LINES+1
 6001 IF(MAP.EQ.0)  GO TO 555
      WRITE (6,72)
   72 FORMAT(1H1,1X,////)
      WRITE (6,724)  KAA
  724 FORMAT(31X,12HPROSPECT NO:,I3///)
      WRITE (6,6002)
 6002 FORMAT (29X, 20H******************** )
      WRITE (6,6003)
 6003 FORMAT (29X, 1H*, 18X, 1H*)
      WRITE (6,6003)
      IF(OPTION.EQ.2)GO TO 1
      WRITE (6,6100)
 6100 FORMAT (29X, 1H*, 3X, 12HOPTIMUM PATH, 3X, 1H*)
      GO TO 3
    1 WRITE(6,2)
    2 FORMAT (29X, 1H*, 3X, 12HOPTION  PATH, 3X, 1H*)
    3 WRITE (6,6200)
 6200 FORMAT (1H+, 32X, 12H____________ )
      WRITE (6,6003)
 6400 FORMAT (29X, 1H*, 5X, A1, I1, 4H TO , A1, I1, 5X, 1H*)
      XB=0.
      DO 6703 I=1,LINE1
      IF(IP.EQ.1)GO TO 6706
      IF((I.EQ.1).AND.(KA.NE.1))GO TO 6704
      IF((I.EQ.LINE1).AND.(KA.NE.IP))GO TO 6705
      GO TO 6706
 6705 KA2=KA+1
      KR=PATH(KA2)
      WRITE(6,6402)KL(IIKEY(I)),JJKEY(I),NB(KR)
 6402 FORMAT (29X, 1H*, 5X, A1, I1, 4H TO ,     A2, 5X, 1H*)
      GO TO 6703
 6704 KA1=KA-1
      KG=PATH(KA1)
      WRITE(6,6401)NB(KG),KL(KKKEY(I)),LLKEY(I)
 6401 FORMAT (29X, 1H*, 5X,     A2, 4H TO , A1, I1, 5X, 1H*)
      WRITE(6,6003)
      XB=XB+B(1,IIKEY(I),JJKEY(I),KKKEY(I),LLKEY(I))
      GO TO 6703
 6706 WRITE(6,6400)KL(IIKEY(I)),JJKEY(I),KL(KKKEY(I)),LLKEY(I)
      WRITE(6,6003)
      XB=XB+B(1,IIKEY(I),JJKEY(I),KKKEY(I),LLKEY(I))
 6703 CONTINUE
      WRITE (6,6600)  XB
 6600 FORMAT (29X, 1H*, 6X, F5.0, 7X, 1H*)
      WRITE (6,6003)
      WRITE (6,6003)
      WRITE (6,6002)
      WRITE (6,72)
      RETURN
C
C
  555 WRITE (6,72)
      WRITE (6,724)  KAA
      WRITE (6,260)
  260 FORMAT(16X, 46H**********************************************)
      WRITE (6,261)
  261 FORMAT(16X, 1H*, 44X, 1H*)
      WRITE (6,261)
      IF(OPTION.EQ.2)GO TO 11
      WRITE (6,262)
  262 FORMAT(16X, 1H*, 10X, 24HOPTIMUM PATH INFORMATION, 10X, 1H*)
      GO TO 23
   11 WRITE(6,12)
   12 FORMAT(16X, 1H*, 10X, 24HOPTION  PATH INFORMATION, 10X, 1H*)
   23 WRITE (6,261)
      WRITE (6,261)
      WRITE (6,263)
  263 FORMAT(16X, 12H*   FROM  TO, 5X, 5HCABLE, 5X, 5HMILES, 5X,
     1 9HHOURS   *)
      WRITE (6,264)
  264 FORMAT  (1H+, 19X, 8H____  __,5X, 5H_____, 5X, 5H_____, 5X, 5H____
     1_)
      WRITE (6,261)
  265 FORMAT(16X, 1H*, 4X,A1,I1,3H - ,A1, I1, 6X, 3H IN, 6X, F4.0, 6X,
     1 F4.0, 4X, 1H*)
  266 FORMAT(16X, 1H*, 4X, A1, I1, 3H - ,   A1, I1, 6X, A3, 6X, F4.0, 6X
     1, F4.0, 4X, 1H*)
  267 FORMAT(16X, 1H*, 4X, A1, I1,3H - ,A1,I1,6X, 3H IN, 6X, F4.0, 6X,
     1 F4.0, 4X, 1H*)
  268 FORMAT(16X, 1H*, 4X, A1, I1, 3H - ,   A1, I1, 6X, 3HOUT, 6X, F4.0,
     1 6X, F4.0, 4X, 1H*)
      WB3=B(3,IIKEY(1),JJKEY(1),KKKEY(1),LLKEY(1))
      WB1=B(1,IIKEY(1),JJKEY(1),KKKEY(1),LLKEY(1))
      IF(IP.EQ.1)GO TO 1500
      IF(KA.EQ.1)GO TO 1500
      KA1=KA-1
      KG=PATH(KA1)
      WRITE(6,1265)NB(KG),KL(KKKEY(1)),LLKEY(1),WB3,WB1
 1265 FORMAT(16X, 1H*, 4X,   A2,3H - ,A1, I1, 6X, 3H IN, 6X, F4.0, 6X,
     1 F4.0, 4X, 1H*)
      GO TO 1501
 1500 WRITE(6,265)KL(IIKEY(1)),JJKEY(1),KL(KKKEY(1)),LLKEY(1),WB3,WB1
 1501 LL=3-LLKEY(1)
      WRITE(6,261)
      I2=KKKEY(1)-1
      WRITE(6,268)KL(KKKEY(1)),LLKEY(1),KL(KKKEY(1)),LL,
     1DLINE(I2),TIME(I2)
      WRITE(6,261)
      DO 500 I=2,LINES
      KBETA=1
      LL=3-LLKEY(I)
      XYZD=B(3,IIKEY(I),JJKEY(I),KKKEY(I),LLKEY(I))-CABLE
      IF(XYZD.GT.BEDIST)KBETA=2
      WB3=B(3,IIKEY(I),JJKEY(I),KKKEY(I),LLKEY(I))
      WB1=B(1,IIKEY(I),JJKEY(I),KKKEY(I),LLKEY(I))
      WRITE(6,266)KL(IIKEY(I)),JJKEY(I),KL(KKKEY(I)),LLKEY(I),
     1 CABPOS(KBETA),WB3,WB1
      WRITE(6,261)
      I1=KKKEY(I)-1
      WRITE(6,268)KL(KKKEY(I)),LLKEY(I),KL(KKKEY(I)),LL,
     1DLINE(I1),TIME(I1)
      WRITE(6,261)
  500 CONTINUE
      J=LINE1
      WB3=B(3,IIKEY(J),JJKEY(J),KKKEY(J),LLKEY(J))
      WB1=B(1,IIKEY(J),JJKEY(J),KKKEY(J),LLKEY(J))
      IF(IP.EQ.1)GO TO 2500
      IF(KA.EQ.IP)GO TO 2500
      KA2=KA+1
      KR=PATH(KA2)
      WRITE(6,1267)KL(IIKEY(J)),JJKEY(J),NB(KR),WB3,WB1
 1267 FORMAT(16X, 1H*, 4X, A1, I1,3H - ,   A2,6X, 3H IN, 6X, F4.0, 6X,
     1 F4.0, 4X, 1H*)
      GO TO 2501
 2500 WRITE(6,267)KL(IIKEY(J)),JJKEY(J),KL(KKKEY(J)),LLKEY(J),
     1WB3,WB1
 2501 YMILES=0.
      YHOURS=0.
      DO 600 I=1,LINE1
      IF(IP.EQ.1)GO TO 2502
      IF((I.EQ.LINE1).AND.(KA.NE.IP))GO TO 600
 2502 YMILES=YMILES+B(3,IIKEY(I),JJKEY(I),KKKEY(I),LLKEY(I))
      YHOURS=YHOURS+B(1,IIKEY(I),JJKEY(I),KKKEY(I),LLKEY(I))
  600 CONTINUE
      WRITE (6,261)
      WRITE (6,261)
      WRITE (6,260)
      WRITE (6,261)
      WRITE (6,261)
      WRITE (6,270)
  270 FORMAT(16X, 1H*, 24X, 5HHOURS, 4X, 5HMILES, 6X, 1H*)
      WRITE (6,271)
  271 FORMAT (1H+, 40X, 5H_____, 4X, 5H_____)
      WRITE (6,261)
      XHOURS = 0.
      XMILES = 0.
      DO 876  K=1,LINES
      XHOURS = XHOURS + TIME(K)
      XMILES = XMILES + DLINE(K)
  876 CONTINUE
      WRITE (6,272)  XHOURS, XMILES
  272 FORMAT(16X, 1H*, 6X, 10HPRODUCTIVE, 8X, F5.0, 4X, F5.0, 6X, 1H*)
      WRITE (6,261)
      WRITE (6,273)  YHOURS, YMILES
  273 FORMAT(16X, 1H*, 6X, 14HNON-PRODUCTIVE, 4X, F5.0, 4X, F5.0, 6X,
     1 1H*)
      WRITE (6,274)
  274 FORMAT (1H+, 40X, 5H_____, 4X, 5H_____)
      WRITE (6,261)
      WHOURS = XHOURS + YHOURS
      WMILES = XMILES + YMILES
      WRITE (6,275)  WHOURS, WMILES
  275 FORMAT(16X, 1H*, 6X, 5HTOTAL, 13X, F5.0, 4X, F5.0, 6X, 1H*)
      WRITE (6,261)
      WRITE (6,261)
      WRITE (6,260)
      WRITE (6,72)
C
C    ***********************************************************************
      RETURN
      END
      SUBROUTINE ROUTE(KAA,TYPE)
C
C     BRANCH BOUND-LITTLE'S METHOD(STRATEGY 2-NEW VERSION)
C
      IMPLICIT INTEGER (A-V,Z)
      COMMON /M1/B(3,15,2,15,2)/M2/IINF(2000),IPROB(2000)
      COMMON /G1/ILINK(15),JLINK(15),KLINK(15),LLINK(15)
      COMMON /S1/M,MM,N,R,S,BOUND/S2/MAX,ICOST,KCOST,LK,TB,GCOST,BCOST
      COMMON /S3/TM,SR
      COMMON /R5/NL(5),PATH(5)
C
C     INITIATE ALL DATA
C
      GO TO (61,62),TYPE
   61 N=NL(KAA)+1
      GO TO 65
   62 N=KAA
   65 TB=M
      MM=2*M
      COST=0
      R=1
      CALL ELAPSE(ITIME)
      JTIME=0
C
C     CALL SUBROUTINE SSTREE TO CALCULATE LOW BOUND USING SHORTEST SPANNING TREE
C     METHOD.
C
      CALL SSTREE
C
C     ADD UP THE PRODUCTIVE DISTANCE AND ASSIGN THE INFINITE
C     TO EACH PRODUCTIVE LINE
C
      DO 9 I=1,N
      IF(I.NE.1)COST=COST+B(1,I,1,I,2)
      B(1,I,1,I,1)=M
      B(1,I,1,I,2)=M
      B(1,I,2,I,1)=M
      B(1,I,2,I,2)=M
    9 CONTINUE
      GCOST=COST
C
C     ASSIGN ROW OF P2 AND COLUMN OF P1 TO INFINITY
C
      B(1,1,2,1,1)=MM
      B(1,1,2,1,2)=MM
C
C     SEARCH THE BRANCH
C
   16 IF(R.GT.1)GO TO 500
      S=1
      GO TO 15
C
C     FIND THE NODE WITH LONGEST INCLUSIVE LINKAGE.
```

```
C
  500 MAX=-M
      E=0
      DO 50 I=2,R
      IG=IINF(I)/1000000000
      IF(IG.EQ.1)GC TO 50
      BV=IPROB(I)/10000
      IF(TB.GT.BV)GO TO 20
      IINF(I)=IINF(I)+1000000000
      GO TO 50
   20 INC=(IINF(I)-(IINF(I)/1000000000)*1000000000)/10000000
      IF(MAX.GE.INC)GO TO 50
      E=I
      MAX=INC
   50 CONTINUE
C
C     IF THERE IS NO NODE WITH BOUND THAT IS LESS THAN CURRENT OPTIMAL
C     SOLUTION, STOP.  OTHERWISE, FIND THE NODE WITH LEAST BOUND.
C
      IF(E.EC.0)RETURN
      BOUND=IPROB(E)/10000
      IINF(E)=IINF(E)+1000000000
      S=E
C
C     RESET THE COST MATRIX FOR THIS NODE TO COORESPOND WITH SOME CONSTRAINTS.
C
   15 CALL MSET
      IF(LK.EQ.N)GO TO 500
C
C     REDUCE THE COST MATRIX, IF NEED.
C
      ICOST=0
      KCOST=0
   10 DO 11 I=1,N
      CALL REDUC1(I)
   11 CONTINUE
      DO 12 K=1,N
      CALL REDUC2(K)
   12 CONTINUE
C
C     FIND THE MAXIMUM EXCLUSIVE LINKAGE THAT WILL BE USED TO BRANCH SUB-NODE.
C
      IF(S.GT.1)GO TO 14
      BOUND=COST+ICOST+KCOST
   14 CALL PIVOT
      CALL ELAPSE(ITIME)
      JTIME=JTIME+ITIME
      IF(JTIME.LT.TM)GO TO 500
      XCOST=BCOST+GCOST
      XTOTAL=TOTAL
      XDIFF=(XTOTAL-XCOST)/XCOST
      IF(XDIFF.GT.SR)GO TO 500
      WRITE(6,255)
  255 FORMAT(1H0,9X,'THE LAST SCLUTION ABOVE IS AN APPROXIMATION')
      RETURN
      END
      SUBROUTINE MSET
C
C
C     SUBROUTE MSET IS USED TO REST THE COST MATRIX FOR SOME PARTICULAR NODE TO
C     COORESPOND WITH SOME CONSTRAINTS.
C
C
      IMPLICIT INTEGER (A-V,Z)
      DIMENSION KL(15)
      COMMON /M1/B(3,15,2,15,2)/M2/IINF(2000),IPROB(2000)
      COMMON /G1/ILINK(15),JLINK(15),KLINK(15),LLINK(15)
      COMMON /V1/IIKEY(15),JJKEY(15),KKKEY(15),LLKEY(15)
      COMMON /S1/M,MM,N,R,S,BOUND/S2/MAX,ICOST,KCOST,LK,TB,GCOST,BCOST
      DATA KL/1HP,1HA,1HB,1HC,1HD,1HE,1HF,1HG,1HH,1HI,1HJ,1HK,1HL,1HM,
     11HN/
C
C     COPY THE ORIGINAL DISTANCE MATRIX TO OPEPATIONAL MATRIX.
C
      LK=0
      DO 1 I=1,N
      DO 1 J=1,2
      DO 1 K=1,N
      DO 1 L=1,2
      B(2,I,J,K,L)=B(1,I,J,K,L)
    1 CONTINUE
C
C     TRACE THE INFORMATION BACK FROM CURRENT NODE UPTO ALL TOUR NODE.
C     UNPACK THE INFORMATION FOR EACH NODE TO OBTAIN LINKAGE AND BOUND VALUE.
C
      IF(S.EQ.1)RETURN
      IS=S
   50 ID=(IINF(IS)-(IINF(IS)/10000000)*10000000)/1000000
    3 IG=IINF(IS)
    5 LL=IG-(IG/10)*10
      KK=(IG-(IG/1000)*1000)/10
      JJ=(IG-(IG/10000)*10000)/1000
      II=(IG-(IG/1000000)*1000000)/10000
C
C     DETERMINE WHETHER IT IS INCLUSIVE LINKAGE OR EXCLUSIVE LINKAGE.
C
      IF(ID.EQ.2)GO TO 10
      LK=LK+1
      ILINK(LK)=II
      JLINK(LK)=JJ
      KLINK(LK)=KK
      LLINK(LK)=LL
      JJJ=3-JJ
      LLL=3-LL
      B(2,II,1,1,1)=MM
      B(2,II,2,1,2)=MM
      B(2,KK,LL,1,LL)=MM
      B(2,1,JJJ,II,JJ)=MM
      B(2,1,1,KK,2)=MM
      B(2,1,2,KK,1)=MM
      GO TO 20
   10 IF(B(2,II,JJ,KK,LL).EQ.MM)GO TO 20
      B(2,II,JJ,KK,LL)=M
   20 IP=IPROB(IS)-(IPROB(IS)/10000)*10000
```

```
      IF(IP.EQ.9999)GO TO 100
      IS=IP
      GO TO 50
C
C     IF  ALL  LINKAGE FORM A CHAIN OF N LINKAKES, THEN PRINT OUT OPTIMAL ROUTE
C     AND RETURN.  OTHERWISE, CALL SUBROUTINE LINK TO FIND THE LINKAGE THAT WILL
C     FORM SUBTOUR AND SET THIS LINKAGE TO INFINITY.
C
  100 IF(LK.EQ.N)GO TO 95
      CALL LINK
      RETURN
   95 IF(TB.LE.BOUND)RETURN
      TB=BOUND
C
C     SEQUENCE ALL LINKAGES TO START WITH P1 AND END WITH P2.
C
      IA=1
      JA=1
      IG=0
   24 IG=IG+1
      IF(IG.GT.LK)GO TO 25
      DO 22  I=1,LK
      IF((IA.EQ.ILINK(I)).AND.(JA.EQ.JLINK(I)))GO TO 23
   22 CONTINUE
   23 IIKEY(IG)=ILINK(I)
      JJKEY(IG)=JLINK(I)
      KKKEY(IG)=KLINK(I)
      LLKEY(IG)=LLINK(I)
      IA=KLINK(I)
      JA=3-LLINK(I)
      ILINK(I)=0
      JLINK(I)=0
      KLINK(I)=0
      LLINK(I)=0
      GO TO 24
C
C     WRITE THE OPTIMAL TOUR
C
   25 WRITE(6,6)(KL(IIKEY(I)),JJKEY(I),KL(KKKEY(I)),LLKEY(I),I=1,LK)
    6 FORMAT(9X,A1,I1,14X,A1,I1)
      QB=TB-GCOST
      WRITE(6,7)TB,QB
    7 FORMAT(1H0,'TOTAL DISTANCE=',I5,6X,'NON-PRODUCTIVE DISTANCE=',I5,
     1/////)
      RETURN
      END
      SUBROUTINE REDUC1(I)
C
C     SUBROUTINE REDUC1(I) IS USED TO FIND ROW MINIMUM IN EACH ROW
C     AND SUBTRACT THE MINIMUM FROM THIS ROW
C
      IMPLICIT INTEGER (A-V,Z)
      COMMON /M1/B(3,15,2,15,2)/M2/IINF(2000),IPROB(2000)
      COMMON /S1/M,MM,N,R,S,BOUND/S2/MAX,ICOST,KCOST,LK,TB,GCOST,BCOST
C
C     FIND ROW MINIMUM
C
      MIN=M
      DO 2 J=1,2
      IF(B(2,I,J,1,J).EQ.MM)GO TO 2
      DO 1 K=1,N
      DO 1 L=1,2
      LL=3-L
      IF(B(2,1,LL,K,L).EQ.MM)GO TO 1
      IF(B(2,I,J,K,L).LT.MIN)MIN=B(2,I,J,K,L)
    1 CONTINUE
    2 CONTINUE
C
C     SUBTRACT ROW MINIMUM FROM EACH ROW
C
      IF(MIN.EQ.0.OR.MIN.EQ.M)RETURN
      ICOST=ICOST+MIN
      DO 4 J=1,2
      IF(B(2,I,J,1,J).EQ.MM)GO TO 4
      DO 3 K=1,N
      DO 3 L=1,2
      LL=3-L
      IF(B(2,1,LL,K,L).EQ.MM)GO TO 3
      IF(B(2,I,J,K,L).EQ.M)GO TO 3
      B(2,I,J,K,L)=B(2,I,J,K,L)-MIN
    3 CONTINUE
    4 CONTINUE
      RETURN
      END
      SUBROUTINE REDUC2(K)
C
C     SUBROUTINE REDUC2(K) IS USED TO FIND CCLUMN MINIMUM IN EACH COLUMN
C     AND SUBTRACT THE MINIMUM FROM THIS COLUMN
C
      IMPLICIT INTEGER (A-V,Z)
      COMMON /M1/B(3,15,2,15,2)/M2/IINF(2000),IPROB(2000)
      COMMON /S1/M,MM,N,R,S,BOUND/S2/MAX,ICOST,KCOST,LK,TB,GCOST,BCOST
C
C     FIND COLUMN MINIMUM
C
      MIN=M
      DO 2 L=1,2
      LL=3-L
      IF(B(2,1,LL,K,L).EC.MM)GO TO 2
      DO 1 I=1,N
      DO 1 J=1,2
      IF(B(2,I,J,1,J).EQ.MM)GO TO 1
      IF(B(2,I,J,K,L).LT.MIN)MIN=B(2,I,J,K,L)
    1 CONTINUE
    2 CONTINUE
C
C     SUBTRACT COLUMN MINIMUM FROM EACH COLUMN
C
      IF(MIN.EQ.0.OR.MIN.EQ.M)RETURN
      KCOST=KCOST+MIN
      DO 4 L=1,2
      LL=3-L
      IF(B(2,1,LL,K,L).EQ.MM)GO TO 4
      DO 3 I=1,N
      DO 3 J=1,2
      IF(B(2,I,J,1,J).EQ.MM)GO TO 3
      IF(B(2,I,J,K,L).EQ.M)GC TO 3
      B(2,I,J,K,L)=B(2,I,J,K,L)-MIN
    3 CONTINUE
    4 CONTINUE
      RETURN
      END
      SUBROUTINE PIVOT
C
C     SUBROUTINE PIVOT IS USED TO FIND THE MAXIMUM EXCLUSIVE LINKAGE
C     THAT WILL BE USED TO BRANCH SUB-NODE.
C
      IMPLICIT INTEGER (A-V,Z)
      DIMENSION SECROW(20),SECCOL(20),IZ(60),JZ(60),KZ(60),LZ(60)
      COMMON /M1/B(3,15,2,15,2)/M2/IINF(2000),IPROB(2000)
      COMMON /G1/ILINK(15),JLINK(15),KLINK(15),LLINK(15)
      COMMON /S1/M,MM,N,R,S,BOUND/S2/MAX,ICOST,KCOST,LK,TB,GCOST,BCOST
      Q=0
C
C     SEARCH FOR ZERO ENTRY IN EACH ROW AND FIND ROW MINIMUM
C
      DO 3 I=1,N
      Q1=0
      SECROW(I)=M
      DO 3 J=1,2
      IF(B(2,I,J,1,J).EQ.MM)GO TO 3
      DO 2 K=1,N
      DO 2 L=1,2
      LL=3-L
      IF(B(2,1,LL,K,L).EQ.MM)GO TO 2
      IF(B(2,I,J,K,L).EQ.0)GO TO 1
      IF(B(2,I,J,K,L).LT.SECROW(I))SECROW(I)=B(2,I,J,K,L)
      GO TO 2
    1 Q=Q+1
      Q1=Q1+1
      IZ(Q)=I
      JZ(Q)=J
      KZ(Q)=K
      LZ(Q)=L
      IF(Q1.EQ.2)SECROW(I)=0
    2 CONTINUE
    3 CONTINUE
C
C     SEARCH FOR ZERO ENTRY IN EACH COLUMN AND FIND COLUMN MINIMUM
C
      DO 9 K=1,N
      SECCOL(K)=M
      Q1=1
      DO 8 L=1,2
      LL=3-L
      IF(B(2,1,LL,K,L).EQ.MM)GO TO 8
      DO 7 I=1,N
      DO 7 J=1,2
      IF(B(2,I,J,1,J).EQ.MM)GO TO 7
      IF(B(2,I,J,K,L).EQ.0)GO TO 4
      IF(B(2,I,J,K,L).LT.SECCOL(K))SECCOL(K)=B(2,I,J,K,L)
      GO TO 7
    4 GO TO (6,5),Q1
    5 SECCOL(K)=0
      GO TO 9
    6 Q1=Q1+1
    7 CONTINUE
    8 CONTINUE
    9 CONTINUE
C
```

```
C      SEARCH THE MAXIMUM EXCLUSIVE LINKGAE.
C
       MAX=-1
       KEY=0
       IF(Q.EQ.0)RETURN
   30  DO 10 J=1,Q
       I=IZ(J)
       K=KZ(J)
       SAVE=SECROW(I)+SECCOL(K)
       IF(MAX.GE.SAVE)GO TO 10
       MAX=SAVE
       KEY=J
   10  CONTINUE
       R=R+1
C
C      BRANCH CURRENT NODE TC TWC SUB-NODE, ONE EXCLUSIVE NODE AND ONE INCLUSIVE
C      NODE OF MAXIMUM EXCLUSIVE LINKAGE.
C
       IINF(R)= 2000000+(IZ(KEY))*10000+(JZ(KEY))*1000+(KZ(KEY))*10+
      1LZ(KEY)+LK*10000000
       IF(S.EG.1)GO TC 15
       IPROB(R)=(BOUND+MAX)*10000+S
       GO TO 20
   15  IPROB(R)=(BOUND+MAX)*10000+9999
   20  R=R+1
       II=IZ(KEY)
       JJ=JZ(KEY)
       KK=KZ(KEY)
       LL=LZ(KEY)
       JJJ=3-JJ
       LLL=3-LL
       B(2,II,1,1,1)=MM
       B(2,II,2,1,2)=MM
       B(2,KK,LL,1,LL)=MM
       B(2,1,JJJ,II,JJ)=MM
       B(2,1,1,KK,2)=MM
       B(2,1,2,KK,1)=MM
       LK=LK+1
       ILINK(LK)=II
       JLINK(LK)=JJ
       KLINK(LK)=KK
       LLINK(LK)=LL
       CALL LINK
   26  ICOST=0
       KCOST=0
       DO 11 I=1,N
       CALL REDUC1(I)
   11  CONTINUE
       DO 12 K=1,N
       CALL REDUC2(K)
   12  CONTINUE
       IINF(R)= 1000000+(IZ(KEY))*10000+(JZ(KEY))*1000+(KZ(KEY))*10+
      1LZ(KEY)+(LK+1)*10000000
       IF(S.EQ.1)GO TO 25
       IPROB(R)=(BOUND+ICOST+KCOST)*10000+S
       RETURN
   25  IPROB(R)=(BOUND+ICOST+KCOST)*10000+9999
       RETURN
       END
      SUBROUTINE LINK
C
C     SUBROUTINE LINK IS USED TO FIND THE LINKAGE THAT WILL CAUSE CURRENT
C     SEGMENT OF CHAIN TO FORM A SUBTOUR. SET THIS LINKAGE TO INFINITY.
C
      IMPLICIT INTEGER (A-V,Z)
      DIMENSION IV(15),JV(15),KV(15),LV(15)
      COMMON /G1/ILINK(15),JLINK(15),KLINK(15),LLINK(15)
      COMMON /M1/B(3,15,2,15,2)/M2/IINF(2000),IPROB(2000)
      COMMON /S1/M,MM,N,R,S,BOUND/S2/MAX,ICOST,KCOST,LK,TB,GCOST,BCOST
      IF(LK.EQ.0)RETURN
      IF(LK.GE.(N-1))RETURN
C
C     COPY ALL LINKAGES TO OPERATIONAL ARRAY.
C
      DO 1 I=1,LK
      IV(I)=ILINK(I)
      JV(I)=JLINK(I)
      KV(I)=KLINK(I)
      LV(I)=LLINK(I)
    1 CONTINUE
C
C     CHECK IS THERE ANOTHER LINKAGE NEEDED TO BE CONNECTED.  IF THERE IS NONE,
C     RETURN.  OTHERWISE SLECT FIRST LINKAGE.
C
   50 DO 2 E=1,LK
      IF(IV(E).EQ.0)GO TO 2
      I=IV(E)
      J=JV(E)
      JJ=3-J
      K=KV(E)
      L=LV(E)
      LL=3-L
      IV(E)=0
      JV(E)=0
      KV(E)=0
      LV(E)=0
      GO TO 5
    2 CONTINUE
      RETURN
C
C     CHECK IS THERE ANOTHER LINKAGE CONNECTED TO THE END OF A CHAIN.
C     IF SO, MAKE NECESSARY CHANGE FOR THE END AND CONTINUE THIS PROCESS.
C
    5 DO 10 F=1,LK
      IF((IV(F).EQ.K).AND.(JV(F).EQ.LL))GO TO 15
   10 CONTINUE
      GO TO 35
   15 K=KV(F)
      L=LV(F)
      LL=3-L
      IV(F)=0
      JV(F)=0
      KV(F)=0
      LV(F)=0
      GO TO 5
C
C     CHECK IS THERE ANOTHER LINKAGE CONNECTED TO THE BEGINING OF A CHAIN.
```

```
C     IF SO, MAKE NECESSARY CHANGE FOR THE BEGINING AND CONTINUE THIS PROCESS.
C
   35 DO 20 Z=1,LK
      IF((KV(Z).EQ.I).AND.(LV(Z).EQ.JJ))GO TO 25
   20 CONTINUE
C
C     SET THE LINKAGE THAT WILL CLOSE A CHAIN TO INFINITY TO AVOID A SUBTOUR.
C
      IF(B(2,K,LL,I,JJ).EQ.MM)GO TO 50
      B(2,K,LL,I,JJ)=M
      GO TO 50
   25 I=IV(Z)
      J=JV(Z)
      JJ=3-J
      IV(Z)=0
      JV(Z)=0
      KV(Z)=0
      LV(Z)=0
      GO TO 35
      END
      SUBROUTINE SSTREE
C
C
C     SUBROUTINE SSTREE IS USED TO CALCULATE LOW BOUND OF THE PROBLEM BY
C     USING SHORTEST SPANNING TREE METHOD.
C
C
      IMPLICIT INTEGER (A-V,Z)
      DIMENSION V(15,2),A(15,2,15,2),ITEMP(30),JTEMP(30),KTEMP(30),
     1LTEMP(30)
      COMMON /M1/B(3,15,2,15,2)/M2/IINF(2000),IPROB(2000)
      COMMON /S1/M,MM,N,R,S,BOUND/S2/MAX,ICOST,KCOST,LK,TB,GCOST,BCOST
      COMMON /S3/TM,SR
C
C     INITIATE ALL DATA.
C
      BCOST=0
      C=0
      T=0
      NM=2*N-1
      DO 11 I=1,N
      DO 11 J=1,2
   11 V(I,J)=0
C
C     COPY THE ORIGINAL COST MATRIX TO OPERATIONAL MATRIX.
C
      DO 9 I=1,N
      DO 9 J=1,2
      DO 9 K=1,N
      DO 9 L=1,2
    9 B(2,I,J,K,L)=B(1,I,J,K,L)
C
C     ADD LARGER NUMBER TO THE ROW OF STARTING PORT AND THE COLUMN OF ENDING
C     PORT.
C
      DO 50 J=1,2
      DO 50 K=1,N
      DO 50 L=1,2
   50 B(2,1,J,K,L)=B(2,1,J,K,L)+M/2
      DO 55 L=1,2
      DO 55 I=1,N
      DO 55 J=1,2
   55 B(2,I,J,1,L)=B(2,I,J,1,L)+M/2
C
C     ASSIGN THE LINKAGE BETWEEN STARTING PORT AND ENDING PORT A INFINITY TO
C     FORM A CHAIN.
C
      B(2,1,1,1,1)=M
      B(2,1,1,1,2)=M
      B(2,1,2,1,1)=M
      B(2,1,2,1,2)=M
C
C     ASSIGN A ZERO TO ALL PRODUCTIVE LINE.
C
      DO 60 I=2,N
      DO 60 J=1,2
      DO 60 K=2,N
      DO 60 L=1,2
      LL=3-L
      IF((I.EQ.K).AND.(J.EQ.L))B(2,I,J,K,L)=M
      IF((I.EQ.K).AND.(J.EQ.LL))B(2,I,J,K,L)=0
   60 CONTINUE
C
C     AT EACH STAGE OF SUBROUTINE, ONE EDGE IS CONSIDERED WHERELY ONE
C     OF FOUR POSSIBLE CONDITIONS WILL ARISE.  IF NEITHE OF VERTICES IS
C     INCLUDED IN A TREE, THIS ECGE IS TAKEN AS NEW TREE ANC ITS EDGE IS
C     NUMBERED BY AN INCREMENTED NUMBER C.  IF ONE VERTEX IS IN A TREE, THE
C     EDGE WILL BE GROWN TO THIS TREE.  IF THE TWO VERTICES ARE IN DIFFERENT
C     TREE, THESE WILL BE GRAFTED INTC A SINGLE TREE BY RENUMBERING THE VERTICES
C     OF THE COMPONENT.  FINALLY, IF BOTH VERTICES ARE IN THE SAME TREE, THE
C     EDGE COMPLETES A CYCLE OF THE GRAPH AND CONSEQUENTLY WILL NOT BE
C     CONSIDERED FURTHER.
C
   10 MIN=2*MM
      DO 1 I=1,N
      DO 1 J=1,2
      DO 1 K=1,N
      DO 1 L=1,2
      IF(MIN.LE.B(2,I,J,K,L))GO TO 1
      MIN=B(2,I,J,K,L)
      II=I
      JJ=J
      KK=K
      LL=L
    1 CONTINUE
   66 IF(V(II,JJ).NE.0)GO TO 5
      IF(V(KK,LL).NE.0)GO TO 15
      C=C+1
      V(II,JJ)=C
      V(KK,LL)=C
      GO TO 2
   15 V(II,JJ)=V(KK,LL)
      GO TO 2
    5 IF(V(KK,LL).NE.0)GO TO 25
      V(KK,LL)=V(II,JJ)
      GO TO 2
   25 IF(V(II,JJ).EQ.V(KK,LL))GO TO 35
      I1=V(II,JJ)
      K1=V(KK,LL)
      DO 40 E=1,N
      DO 40 F=1,2
      IF(V(E,F).EQ.K1)V(E,F)=I1
   40 CONTINUE
    2 T=T+1
      ITEMP(T)=II
      JTEMP(T)=JJ
      KTEMP(T)=KK
      LTEMP(T)=LL
      BCOST=BCOST+B(2,II,JJ,KK,LL)
   35 B(2,II,JJ,KK,LL)=M
      IF(T.LT.NM)GO TO 10
      BCOST=BCOST-M
      WRITE(6,100)BCCST
  100 FORMAT(1H0,9X,'COST OF SHORTEST SPANNING TREE =',I10)
      RETURN
      END
```

```
*************************************************
*                                               *
*                                               *
*  1-SHIPS THROUGH 3-PRCSPECTS RCUTING PROBLEM  *
*                                               *
*                                               *
*************************************************


                    SHIP NO:  1


*************************************************
*                                               *
*                                               *
*               PORT CO-ORDINATES               *
*                                               *
*                                               *
*              _X1_  _Y1_     _X2_  _Y2_        *
*                                               *
*     P         48.   89.       0.   28.        *
*                                               *
*                                               *
*************************************************
*                                               *
*                                               *
*               PROSPECT NO:  1                 *
*                                               *
*                                               *
*              _X1_  _Y1_     _X2_  _Y2_        *
*                                               *
*     A         38.   80.      22.   70.        *
*     B         42.   78.      28.   66.        *
*     C         44.   74.      32.   62.        *
*     D         28.   80.      48.   68.        *
*     E         24.   76.      40.   62.        *
*                                               *
*                                               *
*************************************************
*                                               *
*                                               *
*               PROSPECT NO:  2                 *
*                                               *
*                                               *
*              _X1_  _Y1_     _X2_  _Y2_        *
*                                               *
*     A         38.   50.      28.   38.        *
*     B         42.   50.      34.   34.        *
*     C         46.   48.      40.   30.        *
*     D         30.   50.      44.   30.        *
*                                               *
*                                               *
*************************************************
*                                               *
*                                               *
*               PROSPECT NO:  3                 *
*                                               *
*                                               *
*              _X1_  _Y1_     _X2_  _Y2_        *
*                                               *
*     A         10.   62.       2.   54.        *
*     B         12.   60.       6.   50.        *
*     C         18.   60.      10.   46.        *
*     D          4.   60.      18.   52.        *
*     E.         2.   58.      14.   46.        *
*                                               *
*                                               *
*************************************************
```

```
                    PENALTY MATRIX FOR PROSPECT NO: 1


        P2    A1    A2    B1    B2    C1    C2    D1    D2    E1    E2
     *************************************************************************

P1 *  49    13    32    13    30    16    31    22    21    27    28

A1 *  39  ****    19     4    17     8    19    10    16    15    18

A2 *  33    19  ****    22     7    22    13    12    26     6    20

B1 *  37     4    22  ****    18     4    19    14    12    18    16

B2 *  27    17     7    18  ****    18     6    14    20    11    13

C1 *  33     8    22     4    18  ****    17    17     7    20    13

C2 *  22    19    13    19     6    17  ****    18    17    16     8

D1 *  40    10    12    14    14    17    18  ****    23     6    22

D2 *  29    16    26    12    20     7    17    23  ****    25    10

E1 *  37    15     6    18    11    20    16     6    25  ****    21

E2 *  21    18    20    16    13    13     8    22    10    21  ****

NCTE: P2 IS AN ARTIFICAL ENDING PORT.


                    PROSPECT NO:  1


                 ********************
                 *                  *
                 *                  *
                 *   OPTIMUM_PATH   *
                 *                  *
                 *     P1 TO B1     *
                 *                  *
                 *     B2 TO C2     *
                 *                  *
                 *     C1 TO D2     *
                 *                  *
                 *     D1 TO A1     *
                 *                  *
                 *     A2 TO E1     *
                 *                  *
                 *     E2 TO #2     *
                 *       42.        *
                 *                  *
                 *                  *
                 ********************
```

```
          PENALTY MATRIX FOR PROSPECT NO: 2


       P2    A1    A2    B1    B2    C1    C2    D1    D2
   ********************************************************

P1 *   31    12    27    12    29    15    32    16    32

A1 *   29  ****    16     4    16     8    20     8    21

A2 *   25    16  ****    18     7    21    14    12    18

B1 *   33     4    18  ****    18     4    20    12    20

B2 *   32    16     7    18  ****    18     7    16    11

C1 *   37     8    21     4    18  ****    19    16    18

C2 *   39    20    14    20     7    19  ****    22     4

D1 *   21     8    12    12    16    16    22  ****    24

D2 *   42    21    18    20    11    18     4    24  ****


NOTE: P1 IS AN ARTIFICAL STARTING PORT AND P2 IS AN ARTIFICAL ENDING PORT.



                PROSPECT NO:  2


              ********************
              *                  *
              *                  *
              *   OPTIMUM PATH   *
              *                  *
              *     #1 TO A1     *
              *                  *
              *     A2 TO B2     *
              *                  *
              *     B1 TO C1     *
              *                  *
              *     C2 TO D2     *
              *                  *
              *     D1 TO #3     *
              *       27.        *
              *                  *
              *                  *
              ********************
```

```
                    PENALTY MATRIX FOR PROSPECT NO: 3

        P2     A1     A2     B1     B2     C1     C2     D1     D2     E1     E2
     *********************************************************************

P1 *    37     23     28     21     24     16     20     28     12     29     16

A1 *    35   ****     11      3     13      8     16      6     13      9     16

A2 *    26     11   ****     12      6     17     11      6     16      4     14

B1 *    34      3     12   ****     12      6     14      8     10     10     14

B2 *    23     13      6     12   ****     16      6     10     12      9      9

C1 *    37      8     17      6     16   ****     16     14      8     16     15

C2 *    21     16     11     14      6     16   ****     15     10     14      4

D1 *    32      6      6      8     10     14     15   ****     16      3     17

D2 *    30     13     16     10     12      8     10     16   ****     17      7

E1 *    30      9      4     10      9     16     14      3     17   ****     17

E2 *    23     16     14     14      9     15      4     17      7     17   ****


NCTE: P1 IS AN ARTIFICAL STARTING PCRT.



                    PROSPECT NO:  3

                 ********************
                 *                  *
                 *                  *
                 *   OPTIMUM_PATH   *
                 *                  *
                 *    #2 TO D2      *
                 *                  *
                 *    D1 TO A1      *
                 *                  *
                 *    A2 TO E1      *
                 *                  *
                 *    E2 TO C2      *
                 *                  *
                 *    C1 TO B1      *
                 *                  *
                 *    B2 TO P2      *
                 *                  *
                 *       55.        *
                 *                  *
                 *                  *
                 ********************
```

VITA

Thomas Chih-Hsiung Chen

Candidate for the Degree of

Doctor of Philosophy

Thesis: OPTIMAL ROUTING OF LARGE SCALE MARINE SEISMIC MAPPING OPERATIONS

Major Field: Industrial Engineering and Management

Biographical:

Personal Data: Born in Tainan, Taiwan, Republic of China, September 28, 1944. Son of Mr. and Mrs. Ying-Ho Chen.

Education: Graduated from First Tainan High School, Tainan, Taiwan, in June, 1963; received Bachelor of Science degree in Industrial Engineering from Tunghai University in June, 1968; received Master of Science degree in Industrial Engineering, System Management Engineering and Operation Research from University of Pittsburgh in August, 1971; completed requirements for the Doctor of Philosophy degree at Oklahoma State University in July, 1976.

Professional Experience: Graduate Teaching Assistant, University of Pittsburgh, 1970-1971; Industrial Engineer and Project Coordinator, Manhattan Industrial, Inc., New Jersey, 1971-1973; Graduate Research Assistant, Oklahoma State University, 1973-1976.