

A HEURISTIC APPROACH
TO THE LINE HAUL PROBLEM

by 6408

ARMOND DALE SMITH

B.S., Fort Hays Kansas State College, 1969

A MASTER'S THESIS

submitted in partial fulfillment of the
requirements for the degree

MASTER OF SCIENCE

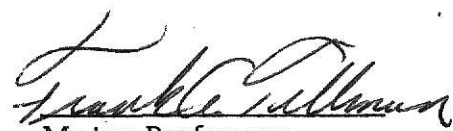
Department of Industrial Engineering

KANSAS STATE UNIVERSITY

Manhattan, Kansas

1971

Approved by


Major Professor

**THIS BOOK
CONTAINS
NUMEROUS PAGES
WITH THE ORIGINAL
PRINTING BEING
SKEWED
DIFFERENTLY FROM
THE TOP OF THE
PAGE TO THE
BOTTOM.**

**THIS IS AS RECEIVED
FROM THE
CUSTOMER.**

**THIS BOOK IS OF
POOR LEGIBILITY
DUE TO LIGHT
PRINTING
THROUGH OUT IT'S
ENTIRETY.**

**THIS IS AS
RECEIVED FROM
THE CUSTOMER.**

LD
2668
74
1971
5553
C.2

TABLE OF CONTENTS

CHAPTER 1		Page
1.1	Introduction	1
1.2	The Line Haul Problem	2
1.3	Literature Survey	3
CHAPTER 2		
2.1	A Heuristic Algorithm	7
2.2	Notation	15
2.3	The Algorithm	16
2.4	Variations on the Algorithm	50
2.5	Experimentation	59
2.6	Conclusions	60
CHAPTER 3		
3.1	Disptaching Routine	64
3.2	Example Problem	67
3.3	Conclusion	101
REFERENCES		
APPENDIX		
A.	Program User's Instructions	105
B.	Computer Program	116
C.	Sample Computer Output	133
D.	Sample Data Set	139

ACKNOWLEDGEMENTS

This research was accomplished through the help and encouragement of many people. A deep gratitude goes to my major professor, Dr. Tillman, for his ideas and help. Also I would like to thank the other members of my committee, Dr. Grosh, Dr. Funk, and Dr. Ashour for reviewing and commenting on this paper.

Special thanks go to my wife, Ivalee, for her encouragement, typing, and putting up with the situation. The typing ability of Norma Freund was much appreciated.

CHAPTER 1

1.1 Introduction

The movement of goods from the place of production to the consumer is a major problem. Goods are moved by trucks, railroads, barges, and planes. In 1968 1,834.3 billion ton miles were required to move the commercial freight of the United States. It is estimated that in 1969 this figure was 1,900.2 billion ton miles. Of this total 21.26% was moved by trucks, 41.04% by railroads, 15.8% by inland waterways, .17% by planes and the remainder was moved by other means. The trucking industry played an even bigger part in intercity freight movement accounting for 51% of the tons of goods shipped and 33% of the ton miles.

In 1969 there were 54,196 commercial fleets in the United States having ten or more vehicles. Of this number, 12,039 were common or contract carriers. In 1968 37,713 million miles were traveled by 871,000 combination truck units for an average of 43,299 miles per unit (1).

Many problems of routing carriers may be categorized into one of the following three types: the traveling salesman problem, the delivery problem, or the line haul problem. The traveling salesman problem involves the visiting of the n cities and returning to the home city. The object is to visit each city once and return home while minimizing the miles traveled.

The delivery problem differs from the traveling salesman problem in that a quantity is either picked up or delivered but where all the cities cannot be served without first returning to the terminal. This is due to some

limitation such as carrier capacity. The delivery problem is a generalization of the traveling salesman problem. The delivery problem is subdivided into single terminal and multi-terminal problems. In the single terminal problem all the cities are served from one terminal. In the multi-terminal case, the cities can be served from one of several terminals. The objective of the delivery problem is to determine the routes which minimize some objective function and which do not violate any system constraints.

The line haul problem is a somewhat different problem in that any city may ship or receive goods from any other city. The goods are shipped as full loads between cities. The objective is to deliver the goods and minimize the empty backhaul miles traveled. This backhaul problem results from the fact that certain cities tend to originate shipments and other cities tend to receive shipments. This creates situations where carriers accumulate at a few points.

The objectives of this study are to first develop a solution technique for the line haul routing problem and second to devise a carrier dispatching routine.

1.2 The Line Haul Problem

As stated previously the line haul problem consists of routing loads to and from n facilities. The number of trips required to move all freight may vary from zero to a number of loads between any two cities. The problem that arises is that the loads the cities ship and receive do not normally balance thus causing a considerable number of empty backhauls.

If partial loads occur they may be left to accumulate to a full load for a later period. Some simplifying assumptions are made to solve this problem. It is assumed that the carriers are similar in performance and have equal capacities. Thus any carrier can be assigned to any load.

The particular algorithm developed to solve the line haul problem provides the following features:

1. Obtains the best routing pattern to minimize the empty backhaul mileage.
2. Efficiently solves large practical problems.
3. Allows the inclusion of system restrictions.
4. Includes previous routes in the present solution computation.
5. Insures that the carriers return to their originating cities.
6. Provides a scheme for dispatching carriers.

1.3 Literature Survey

The line haul problem has received little attention in the literature. The delivery problem including the special case of the traveling salesman situation has been treated extensively in the literature.

The methods used to solve the line haul problem have been mathematical and heuristic programming (6). A number of approaches have been developed for solving the delivery problem. Some of the methods used are dynamic programming (3), integer programming (2), (20), simulation (5), (12),

branch and bound (19), (23), and heuristic programming (7), (8), (11), (14), (15), (16), (24), (25), (26), (27).

The solution of the line haul problem is discussed in the C. E. I. R. report (6). The techniques suggested were linear, integer, and heuristic programming. An example five point problem was solved with each of the techniques.

Linear programming was first used to solve the problem. The objective used was to minimize the distance required to deliver the freight. Possible routes were created by a matrix generator which considered route restrictions. There is a limitation with this approach in that all possible routes and their sequence must be predetermined and included as a variable. This greatly limits the size of the problem which can be solved by this technique. Also the empty backhaul problem was not considered. The program established the combination of routes which best distributed the freight. The solution was not an integer solution, and values were adjusted to the next larger integer.

The same problem was solved using an integer programming code. Results were integers and gave a better system than for the rounded linear solution. Again this method is restricted to very small problems. Their heuristic algorithm was also used to solve the problem. It attempted to minimize the back-tracking and to prevent freight run out at any city if possible. No means was provided for returning carriers to their origin or preventing carriers from concentrating at certain cities.

The routes formed compared favorably with their integer programming solution. These approaches were not very promising in that only small size problems could be solved and backhauls were not considered.

Several methods have been developed to solve the delivery problem. Dynamic programming was applied to the shortest route problem by Bellman (3). This problem was similar to the traveling salesman problem.

An integer programming model of the delivery problem was developed by Balinski and Quendt (2). The cutting plane algorithm was used to determine the optimal solution. The method was limited to problems with few variables and restraints due to the storage and computer time requirements.

A branch and bound approach to the traveling salesman problem was developed by Little and associates (19). Pierce (23) proposed a modification of Little's technique and extended the idea to a single terminal delivery problem. Several constraints were included as delivery time, carrier capacity and other carrier restraints.

Hayes also developed a branch and bound algorithm for the delivery problem which is an extension of Little's work. This method was only successfully applied to problems of twenty points or less.

Application of heuristic programming to the delivery program was first made by Dantzig and Ramser (11). Their algorithm formed single point routes for all demand points. These routes were then combined using the distance saved as the criteria for establishing routes.

This approach was modified by Clarke and Wright (7) to shift

emphasis from vehicle loading to minimization of total distance. Again this algorithm initially created a route from the terminal to each city. Routes were then formed by adding points based on the distance saved. Large practical problems can be solved with this technique.

Cochran (8) modified Clarke and Wright's algorithm to reassign carriers and considered additional restrictions on the systems. This procedure was extended by Tillman and Cochran (26) to include a look ahead feature for forming routes. Hering (16) further investigated the look ahead procedure. It was found that the solution was improved, but at a greatly increased cost of computation time. A multi-terminal algorithm was developed by Tillman (24). It utilizes a savings concept similar to Clark and Wright (7).

Of all the possible methods of solution for the line haul problem dynamic programming, linear programming, and simulation were not suitable for solving large practical problems. Integer programming is impractical because of the lack of efficient computer codes. Because it is possible to solve large practical problems with heuristic techniques, the decision was made to develop a heuristic algorithm using the idea of distance saved to form routes.

CHAPTER 2

2.1 A Heuristic Algorithm

Of all the approaches studied, heuristic techniques seem to be the most promising for obtaining a practical solution to the line haul problem. Heuristic programming may be described as an intuitive method for obtaining a solution to the problem which may or may not give an optimal solution. Such techniques are suitable for complex problems where no analytical approaches are available.

The line haul problem is such a complex problem. Specifically the line haul problem is one where a group of cities may ship and receive commodities to and from each other. This implies that there is a problem of insuring that the carriers permanently assigned to any one city return to that city. Thus the problem is twofold: first insuring that all commodities are delivered, and second that the carriers are returned to their home city. In the second part it would be most efficient to return a carrier to its home city loaded whenever possible. This is not always possible; thus, some are returned empty. This then becomes the basic criteria for the algorithm, that is to deliver all the commodities to their respective cities and minimize the cost or number of miles the carriers traveled empty and return the carriers to their home city. Included in this problem is the maximum time a carrier may spend on a route before returning to its home city. This period is called the planning horizon. The development of the algorithm will be illustrated by finding the solution to an example problem.

A distance network and the demands for a five point problem are

given in Figure 1 and Table 1. It is assumed that the required number of carriers are available.

Observing the load matrix Table 1, it is evident that many possible routes exist. This is especially true if no limits are placed on the length or number of links of a route. Routes could be formed by pairing the loads going in opposite directions such as the loads from A to B, A-B, and from B to A, B-A. In our example two loads must be moved from A to B and three loads from B to A. The tour A-B-A or B-A-B could be formed with two loads on each arch. Continuing with this pairing, this reduces the number of routes left to be scheduled. After this reduction is carried out for the example problem, the resulting routes formed by pairing are given in Table 2 and the remaining loads to be delivered are illustrated in the load matrix, Table 3. This completes the first step of the algorithm.

The number of loads to be routed has been reduced to less than half of the original number. The next step is to combine these remaining loads on routes so that the overall distance traveled is a minimum and the system constraints are not violated. To accomplish this we develop a concept of savings. Initially we assume that each load is delivered and the truck returns empty. This creates an empty backhaul between these points. An example would be the load from A to C. The route A-C-A would be formed with the link C-A being an empty backhaul. Savings in empty backhaul miles can be made by combining loads on routes between several points.

In the example consider the points C and A. Combining point B on the

**THIS BOOK
CONTAINS
NUMEROUS PAGES
WITH DIAGRAMS
THAT ARE CROOKED
COMPARED TO THE
REST OF THE
INFORMATION ON
THE PAGE.**

**THIS IS AS
RECEIVED FROM
CUSTOMER.**

FIGURE 1
DISTANCE NETWORK OF EXAMPLE

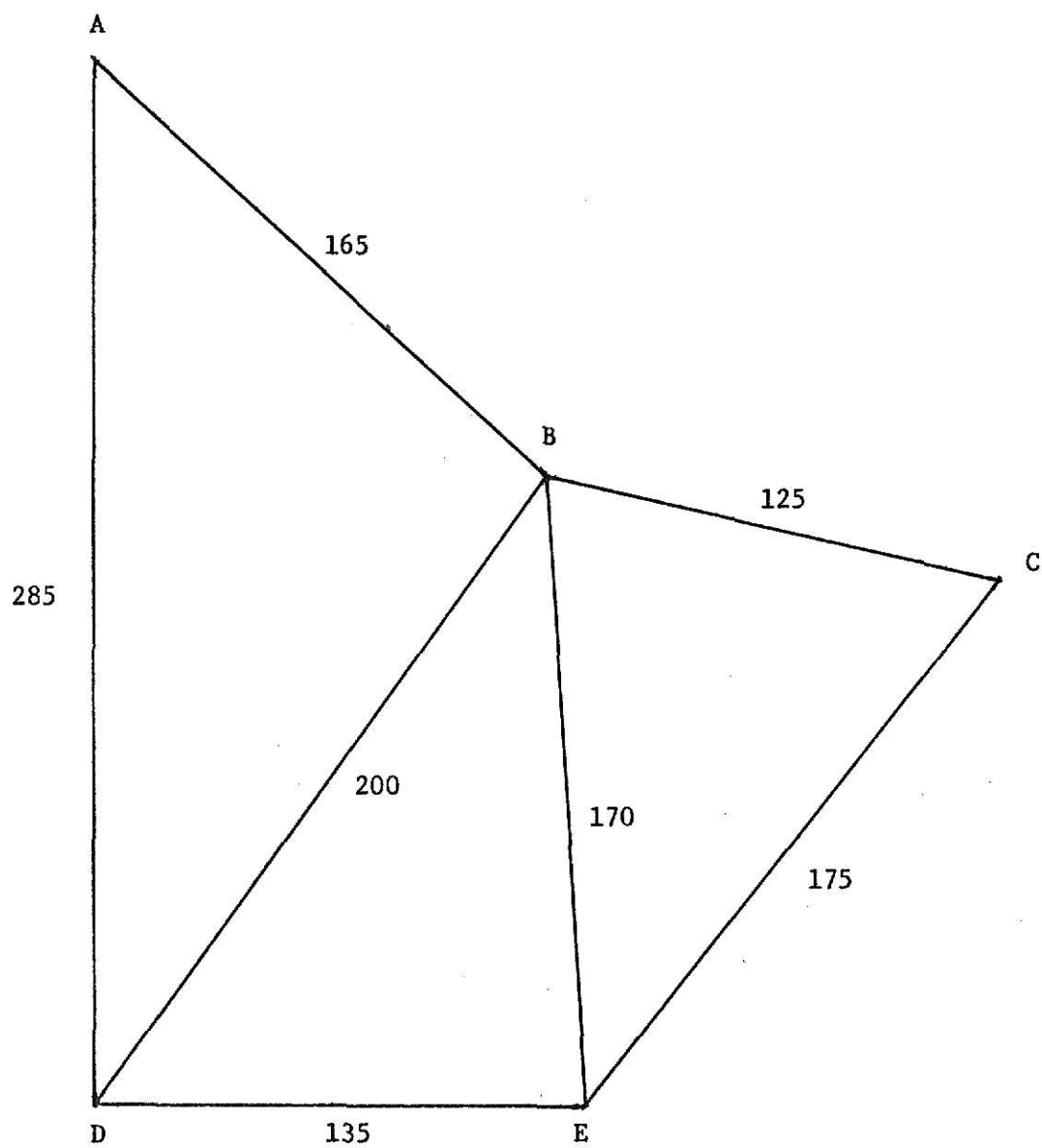


TABLE 1

LOADS TO BE SHIPPED AND RECEIVED

To From	A	B	C	D	E
A	0	2	1	2	0
B	3	0	1	0	1
C	0	2	0	0	2
D	3	0	1	0	2
E	0	1	1	1	0

TABLE 2

ROUTES CREATED BY PAIRING

ROUTE POINTS	NUMBER LOADS	DISTANCE
A-B-A	2	330
A-D-A	2	570
B-C-B	1	250
B-E-B	1	340
C-E-C	1	350
D-E-D	1	270

TABLE 3

LOADS TO BE DELIVERED AFTER PAIRING

To From	A	B	C	D	E
A	0	0	1	0	0
B	1	0	0	0	0
C	0	1	0	0	1
D	1	0	1	0	1
E	0	0	0	0	0

TABLE 4

INITIAL ROUTES

ROUTE POINTS	NUMBER POINTS	DISTANCE
A-C-A	1	580
B-A-B	1	330
C-B-C	1	250
C-E-C	1	350
D-A-D	1	570
D-C-D	1	620
D-E-D	1	270

route between these points forming the tour A-C-B-A, the savings would be computed as follows. Note the distance traveled after the initial pairing is

$$\bar{D} = 2D_{AC} + 2D_{BA} + 2D_{CB} + 2D_{CE} + 2D_{DA} + 2D_{DC} + 2D_{DE}$$

where D_{ij} is the distance between point i and point j . As a result of forming the route A-C-B-A the distance would be

$$\bar{D}' = D_{AC} + D_{CB} + D_{BA} + 2D_{CE} + 2D_{DA} + 2D_{DC} + 2D_{DE}.$$

The difference between the distances \bar{D} and \bar{D}' would be the savings

$$S_{AB}^C = D_{AC} + D_{CB} + D_{BA}$$

eliminating the three empty backhauls. In some situations, only one of these links may be loaded and the other two may be empty backhauls.

In general the savings are calculated using the formula

$$S_{ij}^k = \alpha D_{ki} + \alpha D_{jk} + D_{ij}$$

where the link i, j is the empty backhaul. The parameter α is equal to +1 if there is a load from i to k or k to j and is equal to -1 if there is no load. In the example note that the savings for link C-A through point B resulted from both α 's being equal to +1. Once the points are combined on a route the separate routes such as C-B-C and B-A-B are eliminated from further consideration.

In summary the heuristic algorithm for solving the line haul problem includes the following basic steps:

1. Pair full loads between two points whenever possible.
2. Assume the remaining loads are delivered directly, thus creating a number of empty backhauls.

3. Calculate the potential savings using $S_{ij}^k = \alpha D_{ki} + \alpha D_{jk} + D_{ij}$ for combining links on routes.
4. Form routes which have a $S_{ij}^k > 0$ where the system constraints are not violated.

2.2 Notation

The following notation is used in the subsequent discussion.

D_{ij} = distance between points i and j .

L_{ij} = loads from i to j .

L_s = number of loads to which the savings apply.

P = set of all points in the system.

R or R' = a specific route.

\bar{S}_{ij} = maximum savings of all the current savings.

S_{ij}^k = savings for link $i-j$ at point k .

S_{ij1} = single link savings for link $i-j$.

S_{ij2}^k = two link savings for link $i-j$, through point k .

S_{ij3}^{kh} = three link savings for link $i-j$, through points k and h .

T_{ij} = backhaul loads from i to j .

α = ± 1 , depending upon the value of T_{ij} .

2.3 The Algorithm

The algorithm is primarily composed of the steps stated above in the last section. These steps will be detailed below. Four data matrices are required. These are a load matrix, a backhaul load matrix, a matrix for savings, and a distance matrix. Elements of these matrices are denoted respectively by L_{ij} , T_{ij} , S_{ij}^k , and D_{ij} .

There is a backhaul on link ij to which the savings S_{ij}^k apply. Point k is the point through which the carrier is routed to eliminate the empty backhaul on link ij . The link savings S_{ij}^k is independent of the number of loads hauled. The total savings is the number of loads L_s assigned to the new or updated route times the link savings S_{ij}^k . The determination of the value of L_s is discussed in step 7 below.

A set of routes $\{R\}$ is established. The set originally consists of only two link routes. The following information is kept for each route: the route points, which links have empty backhauls, and the number of loads on the links of the route.

The steps of the algorithm are listed below.

1. Pair the loads to form two link routes for all links where L_{ij} and L_{ji} are both greater than zero. Both links $i-j$ and $j-i$ will be loaded on these routes. The number of loads on the routes will be the minimum of L_{ij} and L_{ji} . Reduce the number of loads L_{ij} and L_{ji} by the minimum of L_{ij} and L_{ji} . This will set either L_{ij} or L_{ji} to zero.

2. Create the backhaul matrix by forming the transpose of the modified load matrix where the elements $T_{ij} = L_{ji}$ for all i and j .
3. Create a set of two link routes $\{R\}$. A route $j-i-j$ is formed for each $T_{ij} > 0$. Each route R has a number of loads corresponding to the number of backhauls T_{ij} . Thus each route has an empty backhaul $i-j$.
4. Calculate a set of savings where for each $T_{ij} > 0$ a savings is computed using $S_{ij}^k = \max_{k \in p} (\alpha D_{ki} + \alpha D_{jk} + D_{ij})$. The parameter α is equal to +1 if T_{ki} or $T_{jk} > 0$ but if T_{ki} or $T_{jk} \leq 0$ then $\alpha = -1$.
5. Find the maximum savings \bar{S}_{ij}^k of the savings matrix. Use an arbitrary rule to break ties between savings. If $\bar{S}_{ij}^k \leq 0$ go to step 11 otherwise continue. Check the value of \bar{S}_{ij}^k to see if it is correct. In some cases this value can be incorrect due to the elimination of backhaul $j-k$ or $k-i$, this results from forming other routes. This is necessary at this point so that all the savings will not have to be recalculated each time a new route is formed. If there is a change in value of S_{ij}^k from \bar{S}_{ij}^k , this step is repeated.
6. The savings \bar{S}_{ij}^k applies to routes which include an empty backhaul link $i-j$. The situation may arise where

- this savings could apply to several previously formed routes passing through the same link $i-j$. If this happens, then arbitrarily select one of the routes. This route may have several loads.
7. The number of loads used in establishing a new route is determined by $L_s = \min [\text{number of empty backhaul loads on link } ij \text{ of route } R; T_{ki} > 0; T_{jk} > 0]$. The "empty backhaul loads on link $i-j$ of route R " occur as a result of a previously formed route. T_{ki} and T_{jk} are the original backhauls after the pairing of loads.
 8. Form a new route R' where link $i-j$ is replaced with links $i-k$ and $k-j$. This new route will have L_s loads. Note that this new route with L_s loads may not entirely replace all of the loads of the old route. When this occurs the old route remains with L_s less loads. In some situations, when T_{ki} or $T_{jk} \leq 0$, empty backhauls will be created as a result of forming the new route. This may happen even though $\bar{S}_{ij}^k > 0$. When this happens, the savings for the empty link $i-k$ or $k-j$ must be calculated since it is possible to create an improved route by eliminating this backhaul.
 9. Reduce the number of loads of the original two link routes $k-j-k$ and $i-k-i$ by L_s . If either of these loads

is reduced to zero, eliminate the route. Only the original backhaul links are used to improve routes and routes having three or more links are not combined.

Update the backhaul matrix by reducing elements

T_{ki} , T_{jk} , and T_{ij} by L_s .

10. As a result of the above step, the savings for link $i-j$ should be recalculated. If $T_{ij} \neq 0$ recalculate S_{ij}^k , however if $T_{ij} = 0$ set $S_{ij}^k = 0$. Return to step 5.
11. If no additional positive savings exist, no further improvement can be made. The set of routes $\{R\}$ are then the solution to the problem.

The backhaul matrix is used to keep track of the loads which can be used to improve routes. This was done because there is a direct correspondence between savings and empty links. This makes it necessary to reverse the row and column for calculating α , updating the backhaul matrix, and reducing the unloaded two link routes. In these cases the needed information is the actual load information.

When ties occur, a decision must be made in steps 5 and 7. It is felt that the individual situation would best determine how these decisions should be made. In step 5, when several equal savings were encountered, the first one was selected. In step 7 the rule used was to choose the route with the largest number of loads so that the largest total savings would be accrued.

Information is stored concerning the magnitude of the savings, links corresponding to the savings, and whether the links are loaded. Likewise, information concerning the points on the route, route loads, and the empty links are stored.

Sample Problem

In order to illustrate the algorithm a ten point problem is solved. The ten point network is illustrated in Figure 2. The load matrix is given in Table 5. Table 6 illustrates the distance matrix which is symmetrical. No restrictions on the system are considered and an unlimited number of carriers are available at each point. The example is solved and each step is illustrated below.

Step (1) Two link routes are formed where loads may go in both directions between two points. That is

$L_{AB} = 1$ and $L_{BA} = 2$, a route A-B-A is formed with both links having one load. The values of L_{AB} and L_{CA} are also paired forming route A-C-A with 2 loads.

Values of both L_{AC} and L_{CA} are reduced to 0. This is continued for all pairs of points resulting in the modified load matrix given in Table 7. The routes formed are displayed in Table 8.

Step (2) The backhaul matrix is now formed by transposing the reduced load matrix. The result is contained in Table 9.

Step (3) Create the initial two link routes. These routes are

NETWORK FOR THE 10 POINT EXAMPLE PROBLEM

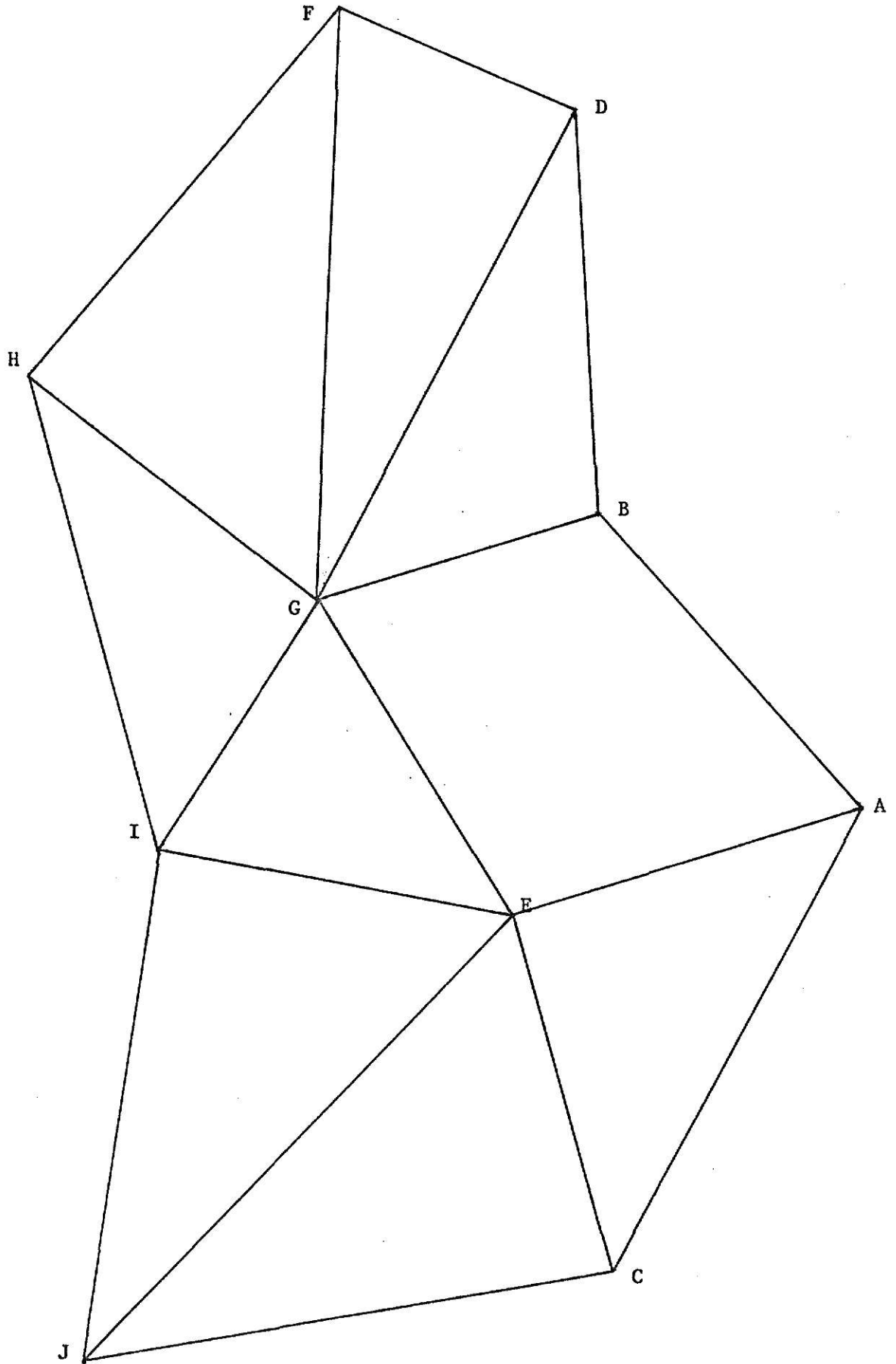


TABLE 5

FREIGHT IN LOADS FOR 10 POINT EXAMPLE PROBLEM

To From	A	B	C	D	E	F	G	H	I	J
A	0	1	2	2	5	2	1	2	4	5
B	2	0	4	1	2	3	5	4	6	4
C	2	3	0	2	3	5	2	2	3	2
D	2	5	5	0	6	7	2	7	5	1
E	4	4	1	2	0	0	4	2	1	3
F	7	6	6	2	2	0	6	3	6	2
G	3	2	3	4	2	4	0	4	3	5
H	5	4	2	1	3	2	5	0	2	6
I	2	2	4	5	3	3	7	6	0	4
J	3	2	2	7	0	2	7	4	3	0

TABLE 6

DISTANCE MATRIX FOR 10 POINT EXAMPLE PROBLEM

To From	A	B	C	D	E	F	G	H	I	J
A	0	85	110	165	80	220	155	235	155	215
B	85	0	195	80	150	135	70	150	130	250
C	110	195	0	270	80	290	160	240	155	120
D	165	80	270	0	190	55	110	160	170	290
E	80	150	80	190	0	210	80	160	75	135
F	220	135	290	55	210	0	130	105	190	310
G	155	70	160	110	80	130	0	80	60	180
H	235	150	240	160	160	105	80	0	105	225
I	155	130	155	170	75	190	60	105	0	120
J	215	250	120	290	135	310	180	225	120	0

TABLE 7

MODIFIED LOAD MATRIX FOR 10 POINT EXAMPLE PROBLEM

To From	A	B	C	D	E	F	G	H	I	J
A	0	0	0	0	1	0	0	0	2	2
B	1	0	1	0	0	0	3	0	4	2
C	0	0	0	0	2	0	0	0	0	0
D	0	4	3	0	4	5	0	6	0	0
E	0	2	0	0	0	0	2	0	0	3
F	5	3	1	0	2	0	2	1	3	0
G	2	0	1	2	0	0	0	0	0	0
H	3	0	0	0	1	0	1	0	0	2
I	0	0	1	0	2	0	4	4	0	1
J	0	0	0	6	0	0	2	0	0	0

TABLE 8

ROUTES FORMED BY PAIRING FOR 10 POINT EXAMPLE PROBLEM

ROUTE NUMBER	ROUTE POINTS	LOADS	DISTANCE	UNLOADED LINK
1	B-A-B	1	170	0
2	C-A-C	2	220	0
3	D-A-D	2	330	0
4	E-A-E	4	160	0
5	F-A-F	2	440	0
6	G-A-G	1	310	0
7	H-A-H	2	470	0
8	I-A-I	2	310	0
9	J-A-J	3	430	0
10	C-B-C	3	390	0
11	D-B-D	1	160	0
12	F-B-E	2	300	0
13	F-B-F	3	270	0
14	G-B-G	2	140	0
15	H-B-H	4	300	0
16	I-B-I	2	260	0
17	J-B-J	2	500	0
18	D-C-D	2	540	0
19	E-C-E	1	160	0
20	F-C-F	5	580	0
21	G-C-G	2	320	0
22	H-C-H	2	480	0
23	I-C-I	3	310	0
24	J-C-J	2	240	0
25	E-D-E	2	380	0
26	F-D-F	2	110	0
27	G-D-G	2	220	0
28	H-D-H	1	320	0
29	I-D-I	5	340	0
30	J-D-J	1	580	0
31	G-E-G	2	160	0
32	H-E-H	2	320	0
33	I-E-I	1	150	0
34	G-F-G	4	260	0
35	H-F-H	2	210	0
36	I-F-I	3	380	0
37	J-F-J	2	620	0

TABLE 8, cont'd.

ROUTE NUMBER	ROUTE POINTS	LOADS	DISTANCE	UNLOADED LINK
38	H-G-H	4	160	0
39	I-G-I	3	120	0
40	J-G-J	5	360	0
41	I-H-I	2	210	0
42	J-H-J	4	450	0
43	J-I-J	3	240	0

TABLE 9

BACKHAUL MATRIX FOR 10 POINT EXAMPLE PROBLEM

To From	A	B	C	D	E	F	G	H	I	J
A	0	1	0	0	0	5	2	3	0	0
B	0	0	0	4	2	3	0	0	0	0
C	0	1	0	3	0	1	1	0	1	0
D	0	0	0	0	0	0	2	0	0	6
E	1	0	2	4	0	2	0	1	2	0
F	0	0	0	5	0	0	0	0	0	0
G	0	3	0	0	2	2	0	1	4	2
H	0	0	0	6	0	1	0	0	4	0
I	2	4	0	0	0	3	0	0	0	0
J	2	2	0	0	3	0	0	2	1	0

TABLE 10
10 POINT EXAMPLE PROBLEM
INITIAL TWO LINK ROUTES FROM STEP 3

ROUTE NUMBER	ROUTE POINTS	LOADS	DISTANCE	UNLOADED LINK
44	B-A-B	1	170	A-B
45	F-A-F	5	440	A-F
46	G-A-G	2	310	A-G
47	H-A-H	3	470	A-H
48	D-B-D	4	160	B-D
49	E-B-E	2	300	B-E
50	F-B-F	3	270	B-F
51	B-C-B	1	390	C-B
52	D-C-D	3	540	C-D
53	F-C-F	1	580	C-F
54	G-C-G	1	320	C-G
55	I-C-I	1	310	C-I
56	G-D-G	2	220	D-G
57	J-D-J	6	580	D-J
58	A-E-A	1	160	E-A
59	C-E-C	2	160	E-C
60	D-E-D	4	380	E-D
61	F-E-F	2	420	E-F
62	H-E-H	1	320	E-H
63	I-E-I	2	150	E-I
64	D-F-D	5	110	F-D
65	B-G-B	3	140	G-B
66	E-G-E	2	160	G-E
67	F-G-F	2	260	G-F
68	H-G-H	1	160	G-H
69	I-G-I	4	120	G-I
70	J-G-J	2	360	G-J
71	D-H-D	6	320	H-D
72	F-H-F	1	210	H-F
73	I-H-I	4	210	H-I
74	A-I-A	2	310	I-A
75	B-I-B	4	260	I-B
76	F-I-F	3	380	I-F
77	A-J-A	2	430	J-A
78	B-J-B	2	500	J-B
79	E-J-E	3	270	J-E
80	H-J-H	2	450	J-H
81	I-J-I	1	240	J-I

given in Table 10.

Step (4) Savings are now calculated for each backhaul link.

For example the savings for eliminating the backhaul from A to B through point C would be $S_{AB}^C = \alpha D_{CA} + \alpha D_{BC} + D_{AB} = -1(110) -1(195) + 85 = -220$. The parameter α is $= -1$ in both cases since there is no backhaul from A to C or C to B. The maximum savings would be found by calculating the savings through all points D, E, F, G, H, I, and J and picking the maximum one. These computations are as follows:

$$S_{AB}^D = \alpha D_{DA} + \alpha D_{BD} + D_{AB} = -1(165) + 1(80) + 85 = 0$$

$$S_{AB}^E = \alpha D_{EA} + \alpha D_{BE} + D_{AB} = +1(80) + 1(150) + 85 = 315$$

$$S_{AB}^F = \alpha D_{FA} + \alpha D_{BF} + D_{AB} = -1(220) + 1(135) + 85 = 0$$

$$S_{AB}^G = \alpha D_{GA} + \alpha D_{BG} + D_{AB} = -1(155) -1(70) + 85 = -140$$

$$S_{AB}^H = \alpha D_{HA} + \alpha D_{BH} + D_{AB} = -1(235) -1(150) + 85 = -300$$

$$S_{AB}^I = \alpha D_{IA} + \alpha D_{BI} + D_{AB} = +1(155) -1(130) + 85 = 110$$

$$S_{AB}^J = \alpha D_{JA} + \alpha D_{BJ} + D_{AB} = +1(215) -1(250) + 85 = 50.$$

The savings through point E results in the maximum savings.

This process is continued for all backhauls and the savings are entered in the savings matrix which appears as Table 11.

In each cell of the table, the point through which the savings are generated is in the upper left corner and the savings is in the lower portion of the cell.

- Step (5) The maximum savings is 675. This savings occurs for S_{DJ}^H , S_{HD}^J , and S_{JH}^D . Since S_{DJ}^H is first, it is selected. A check on S_{DJ}^k is made and the same value is obtained, thus $\bar{S}_{DJ}^H = 675$.
- Step (6) Route number 57 in Table 10 is the only route with an empty backhaul on link D-J, thus it is selected.
- Step (7) Let $L_S = \text{minimum (route loads on } R_{57}, T_{HD}, T_{JH}) = (6, 6, 2) = 2$.
- Step (8) Form a new route R' , numbered 82. It has the points J-D-H-J, with 2 loads, and all other links loaded. The number of loads on route 57, J-D-J, is decreased to 4.
- Step (9) T_{HD} , T_{JH} , and T_{DJ} of the backhaul matrix Table 9 are updated. $T_{HD} = 6 - 2 = 4$, $T_{JH} = 2 - 2 = 0$, $T_{DJ} = 6 - 2 = 4$. The loads on the two link routes H-D-H and J-H-J are reduced by $L_S = 2$. These routes are numbered 71 and 80 and are equal to 4 and 0 loads after the reduction. Route 80 is eliminated from further consideration.
- Step (10) Calculate S_{DJ}^k since $T_{DJ} = 4$, thus the new link savings which passes through point B is $S_{DJ}^B = 620$. Return to Step 5 and continue.

This completes the first iteration of the technique. Two additional iterations are carried out in a similar detailed manner to illustrate how the procedure works.

TABLE II

INITIAL SAVINGS MATRIX FOR 10 POINT EXAMPLE PROBLEM

To From	A	B	C	D	E	F	G	H	I	J
A	0	E 315	0	0	0	I 185	J 550	I 495	0	0
B	0	0	0	J 620	C 425	D 110	0	0	0	0
C	0	E 425	0	J 440	0	E 160	E 320	0	A 200	0
D	0	0	0	0	0	0	E 380	0	0	H 675
E	B 315	0	B 425	J 615	0	B 225	0	I 190	B 355	0
F	0	0	0	G 295	0	0	0	0	0	0
G	0	D 260	0	0	D 380	D 295	0	D 350	A 370	A 550
H	0	0	0	J 675	0	A 120	0	0	A 495	0
I	H 495	E 355	0	0	0	H 190	0	0	0	0
J	G 550	D 620	0	0	D 615	0	0	D 675	D 240	0

- Step (5) A new maximum savings is picked. The maximum value picked is $S_{HD}^I = 675$ but rechecking gives an actual value of $S_{HD}^G = 350$. The next highest savings is $S_{JH}^D = 675$ again recalculating gives a value of $S_{JH}^k = 0$. The next highest savings is $\bar{S}_{BD}^I = 620$. It is recalculated and is determined to be correct, consequently it is selected.
- Step (6) Route number 48 is the only route with an empty backhaul on the link B-D.
- Step (7) $L_S = \min(\text{loads on link BD of } R_{48}, T_{JB}, T_{DJ}) = (4, 2, 4) = 2$.
- Step (8) A new route R' numbered 83 is created with 2 loads and traverses the points D-B-J-D, with no backhaul links. Loads on the old route R_{48} are reduced from 4 to 2.
- Step (9) Loads on routes 78 and 57 are reduced to 0 and 2 respectively. Route 78 is eliminated. $T_{JB} = 2 - 2 = 0$, $T_{DJ} = 4 - 2 = 2$, and $T_{BD} = 4 - 2 = 2$.
- Step (10) Recalculation of S_{BD}^k gives $S_{BD}^G = 260$. Return to Step 5 and continue since other $S_{ij}^k > 0$.

This completes the second iteration.

- Step (5) The maximum savings found are $S_{DJ}^B = 620$ and $S_{JB}^D = 620$. Rechecking these values indicates that $S_{DJ}^E = 615$ and $S_{JB}^k = 0$. The next highest savings are $S_{DJ}^E = S_{ED}^I = S_{DJ}^E = 615$, S_{DJ}^E was checked and selected.
- Step (6) The only route with an empty backhaul on link DJ is 57, J-D-J.

Step (7) $L_S = \min (\text{loads on } R_{57}, T_{ED}, T_{JE}) = (2, 4, 3) = 2.$

Step (8) Route 57 is updated by adding point E. The route is now J-D-E-J with 2 loads and all links are loaded.

Step (9) The following quantities are computed $T_{ED} = 4 - 2 = 2,$
 $T_{JE} = 3 - 2 = 1,$ and $T_{DJ} = 2 - 2 = 0.$ The loads on routes 60 and 79 are reduced by $L_S = 2$ to 2 and 1 respectively.

Step (10) $T_{DJ} = 0$ so S_{DJ}^k is set = 0. Return to Step 5.

Three iterations have been completed. Nineteen more iterations are required and are summarized in Table 15. The resulting savings and back-haul matrices after three iterations are presented in Tables 12 and 13. Routes which were changed and their present form are given in Table 14.

The final routes are listed in Table 16. A total of 89 different tours were created; however, 17 initial tours were eliminated through the modification of routes. These routes had a total distance of 53,430 miles required, of this distance 4,980 miles were traveled empty, 48,450 were traveled loaded.

TABLE 12

SAVINGS MATRIX AFTER THREE ITERATIONS FOR 10 POINT EXAMPLE PROBLEM

To From	A	B	C	D	E	F	G	H	I	J
A	0	E 315	0	0	0	I 185	J 550	I 495	0	0
B	0	0	0	G 260	C 425	D 110	0	0	0	0
C	0	E 425	0	J 440	0	E 160	E 320	0	A 200	0
D	0	0	0	0	0	0	E 380	0	0	0
E	B 315	0	B 425	J 615	0	B 225	0	I 190	B 355	0
F	0	0	0	G 295	0	0	0	0	0	0
G	0	D 260	0	0	D 380	D 295	0	D 350	A 370	A 550
H	0	0	0	G 350	0	A 120	0	0	A 495	0
I	H 495	E 355	0	0	0	H 190	0	0	0	0
J	G 550	0	0	0	D 615	0	0	0	D 240	0

TABLE 13

BACKHAUL MATRIX AFTER THREE ITERATIONS FOR 10 POINT EXAMPLE PROBLEM

To From	A	B	C	D	E	F	G	H	I	J
A	0	1	0	0	0	5	2	3	0	0
B	0	0	0	2	2	3	0	0	0	0
C	0	1	0	3	0	1	1	0	1	0
D	0	0	0	0	0	0	2	0	0	0
E	1	0	2	2	0	2	0	1	2	0
F	0	0	0	5	0	0	0	0	0	0
G	0	3	0	0	2	2	0	1	4	2
H	0	0	0	4	0	1	0	0	4	0
I	2	4	0	0	0	3	0	0	0	0
J	2	0	0	0	1	0	0	0	1	0

TABLE 14

ROUTES CHANGED DURING ITERATIONS 1-3
10 POINT EXAMPLE PROBLEM

ROUTE NUMBER	ROUTE POINTS	LOADS	DISTANCE	UNLOADED LINK
48	D-B-D	2	160	B-D
57	J-D-E-J	2	615	0
60	D-E-D	2	380	E-D
71	D-H-D	4	320	H-D
78	ELIMINATED			
79	E-J-E	1	270	J-E
80	ELIMINATED			
82	J-D-H-J	2	675	0
83	D-B-J-D	2	620	0

TABLE 15

10 POINT EXAMPLE PROBLEM ITERATIONS 4 THROUGH 22

<u>STEP</u>	<u>ACTION</u>
5	<p>Let \bar{S}_{ij}^k be the maximum savings,</p> <p>$\bar{S}_{ij}^k = S_{ED}^J = 615$, checking gives $S_{ED}^k = S_{ED}^G = 380$,</p> <p>$\bar{S}_{ij}^k = S_{JE}^D = 615$, checking gives $S_{JE}^k = S_{JE}^G = 235$,</p> <p>$\bar{S}_{ij}^k = S_{AG}^J = 550$, checking gives the same value.</p>
6	$R = \text{route } 46$.
7	$L_S = \min(2, 2, 2) = 2$.
8	R_{46} changed to G-A-J-G with 2 load and no backhauls.
9	<p>$T_{JA} = 2 - 2 = 0$, $T_{GJ} = 2 - 2 = 0$, $T_{AG} = 2 - 2 = 0$; loads on routes 77 and 70 are reduced by 2 resulting in both being reduced to 0 and eliminated.</p>
10	$T_{AG} = 0 \therefore S_{AG}^k = 0$, go to Step 5.
5	<p>$\bar{S}_{ij}^k = S_{GJ}^A = 550$, checking gives $S_{GJ}^k = S_{GJ}^I = 240$,</p> <p>$\bar{S}_{ij}^k = S_{JA}^G = 550$, checking gives $S_{JA}^k = S_{JA}^H = 225$,</p> <p>$\bar{S}_{ij}^k = S_{AH}^I = 495$, same values on recalculation, $L_S = 2$.</p> <p>$S_{AH}^I = 495 > 0$.</p>
6	$R = \text{route } 47$.
7	$L_S = \min(3, 2, 4) = 2$.
8	Create route R_{84} with points H-A-I-H, loads on R_{84} are equal to 2 with no empty backhaul links, reduce loads

TABLE 15 (CONTINUED)

<u>STEP</u>	<u>ACTION</u>
8	on R_{47} from 3 to 1.
9	$T_{IA} = 2 - 2 = 0$, $T_{HI} = 4 - 2 = 2$, $T_{AH} = 3 - 2 = 1$, reduce loads on R_{74} from 2 to 0 and R_{73} from 4 to 2, eliminate R_{74} .
10	$\bar{S}_{AH}^k = S_{AH}^D = 230$ as $T_{AH} = 1$, go to Step 5.
5	$\bar{S}_{ij}^k = S_{HI}^A = 495$, checking gives $S_{HI}^k = S_{HI}^E = 200$, $\bar{S}_{ij}^k = S_{IA}^H = 495$, checking gives $S_{IA}^k = 0$, $\bar{S}_{ij}^k = S_{CD}^J = 440$, checking gives $S_{CD}^k = S_{CD}^G = 220$, $\bar{S}_{ij}^k = S_{BE}^C = 425$, checking gives the same value.
6	$R = \text{route } 49$.
7	$L_S = \min(2, 1, 2) = 1$.
8	Create route 85 with points E-B-C-E, the number of loads on R_{85} is 1, all links are loaded, reduce the loads on R_{49} to 1.
9	$T_{CB} = 1 - 1 = 0$, $T_{EC} = 2 - 1 = 1$, $T_{BE} = 2 - 1 = 1$, loads on R_{51} and R_{59} are reduced by L_S to 0 and 1, R_{51} is eliminated.
10	$\bar{S}_{BE}^k = S_{BE}^I = 355$, return to Step 5.
5	$\bar{S}_{ij}^k = S_{CB}^E = 425$, checking gives $S_{CB}^k = 0$, $\bar{S}_{ij}^k = S_{EC}^B = 425$, checking gives $S_{EC}^G = 320$. $\bar{S}_{ij}^k = S_{DG}^E = 380$, checking gives the same value.

TABLE 15 (CONTINUED)

<u>STEP</u>	<u>ACTION</u>
6	$R = \text{route } 56.$
7	$L_S = \min (2, 2, 2) = 2.$
8	R_{56} is modified to have points G-D-E-G, two loads and no empty backhaul links.
9	$T_{ED} = 2 - 2 = 0$, $T_{GE} = 2 - 2 = 0$, $T_{DG} = 2 - 2 = 0$, both R_{60} and R_{66} are eliminated since the number of loads are reduced to 0.
10	$\bar{S}_{DG}^k = 0$ as $T_{DG} = 0$, go to Step 5.
5	$\bar{S}_{ij}^k = S_{ED}^I = 380$, checking gives $S_{ED}^k = 0$, $\bar{S}_{ij}^k = S_{GE}^D = 380$, checking gives $S_{GE}^k = 0$, $\bar{S}_{ij}^k = S_{GI}^A = 370$, checking gives $S_{GI}^k = S_{GI}^B = 60$, $\bar{S}_{ij}^k = S_{BE}^I = 355$, checking gives same value.
6	$R = \text{route } 49.$
7	$L_S = \min (1, 4, 2) = 1.$
8	R_{49} is updated to have points E-B-I-E, one load and all links loaded.
9	$T_{IB} = 4 - 1 = 3$, $T_{EI} = 2 - 1 = 1$, $T_{BE} = 1 - 1 = 0$ load on R_{75} are reduced from 4 to 3 and loads on R_{63} from 2 to 1.
10	$T_{BE} = 0$ so $S_{BE}^k = 0$, return to Step 5.

TABLE 15 (CONTINUED)

STEP	ACTION
5	$\bar{S}_{ij}^k = S_{EI}^B = 355$, checking gives $S_{EI}^k = S_{EI}^J = 185$, $\bar{S}_{ij}^k = S_{IB}^E = 355$, checking gives $S_{IB}^k = S_{IB}^G = 120$, $\bar{S}_{ij}^k = S_{GH}^D = 350$, checking gives $S_{GH}^k = S_{GH}^D = 130$, $\bar{S}_{ij}^k = S_{HD}^G = 350$, checking gives $S_{HD}^k = S_{HD}^A = 230$, $\bar{S}_{ij}^k = S_{CG}^E = 320$, checking gives $S_{CG}^k = S_{CG}^J = 220$, $\bar{S}_{ij}^k = S_{EC}^G = 320$, checking gives $S_{EC}^k = S_{EC}^D = 160$, $\bar{S}_{ij}^k = S_{GE}^C = 320$, checking gives $S_{GE}^k = 0$, $\bar{S}_{ij}^k = S_{AB}^E = 315$, checking gives $S_{AB}^k = S_{AB}^E = 15$, $\bar{S}_{ij}^k = S_{EA}^B = 315$, checking gives $S_{EA}^k = S_{EA}^F = 90$, $\bar{S}_{ij}^k = S_{FD}^G = 295$, checking gives $S_{FD}^k = S_{FD}^A = 110$, $\bar{S}_{ij}^k = S_{GF}^D = 295$, checking gives $S_{GF}^k = S_{GF}^D = 75$, $\bar{S}_{ij}^k = S_{BD}^G = 260$, checking gives $S_{BD}^k = S_{BD}^G = 40$, $\bar{S}_{ij}^k = S_{ED}^I = 240$, checking gives $S_{ED}^k = 0$, $\bar{S}_{ij}^k = S_{GE}^D = 240$, checking gives $S_{GE}^k = 0$, $\bar{S}_{ij}^k = S_{JE}^G = 235$, checking gives $S_{JE}^k = S_{JE}^C = 95$, $\bar{S}_{ij}^k = S_{AH}^D = 230$, checking gives the same value.
6	R = route 47.
7	$L_s = \min(1, 4, -) = 1$.
8	R_{47} is incremented by point D to H-A-D-H with one load and link AD empty. It is necessary to calculate S_{AD}^k

TABLE 15 (CONTINUED)

<u>STEP</u>	<u>ACTION</u>
8	which is $S_{AD}^E = 55$.
9	$T_{HD} = 4 - 1 = 3$, $T_{DA} = 0 - 1 = -1$, $T_{AH} = 1 - 1 = 0$, loads on R_{71} are reduced to 3, the two link route having backhaul link DA has been previously eliminated.
10	$S_{AH}^k = 0$, go to Step 5.
5	$\bar{S}_{ij}^k = S_{HD}^A = 230$, checking gives $S_{HD}^k = S_{HD}^E = 130$, $\bar{S}_{ij}^k = S_{EF}^B = 225$, checking gives $S_{EF}^k = S_{EF}^D = 75$, $\bar{S}_{ij}^k = S_{JA}^H = 225$, checking gives $S_{JA}^k = 0$, $\bar{S}_{ij}^k = S_{CD}^G = 220$, checking gives $S_{CD}^k = S_{CD}^E = 160$, $\bar{S}_{ij}^k = S_{CI}^A = 200$, checking gives $S_{CI}^k = S_{CI}^E = 160$, $S_{ij}^k = S_{EH}^I = 190$, this is the value calculated on checking.
6	$R = \text{route } 62$.
7	$L_S = \min(1, -, 2) = 1$.
8	R_{62} is updated to include points H-E-I-H, one load and empty link EI, S_{EI}^k is calculated $S_{EI}^k = S_{EI}^I = 185$.
9	$T_{IE} = 0 - 1 = -1$, $T_{HI} = 2 - 1 = 1$, $T_{EH} = 1 - 1 = 0$, loads on R_{75} are reduced from 2 to 1.
10	$S_{EH}^k = 0$ as $T_{EH} = 0$ go to Step 5.
5	$\bar{S}_{ij}^k = S_{EH}^I = 190$, checking gives $S_{EH}^k = 0$, $\bar{S}_{ij}^k = S_{HI}^F = 190$, checking gives the same value.

TABLE 15 (CONTINUED)

<u>STEP</u>	<u>ACTION</u>
6	R = route 73.
7	$L_S = \min (1, -, 2) = 1.$
8	R_{73} is updated to have points I-H-F-I, one load, and unloaded link H-F, calculate $S_{HF}^k = 0.$
9	$T_{FH} = 0 - 1 = -1, T_{IF} = 3 - 1 = 2, T_{HI} = 1 - 1 = 0,$ R_{76} now has 2 loads.
10	$S_{HI}^k = 0,$ go to Step 5.
5	$\bar{S}_{ij}^k = S_{HI}^F = 190,$ checking gives $S_{HI}^k = S_{HI}^G = 125,$ $\bar{S}_{ij}^k = S_{IF}^H = 190,$ checking gives $S_{IF}^k = S_{IF}^G = 120,$ $\bar{S}_{ij}^k = S_{AF}^I = 185,$ checking gives $S_{AF}^k = S_{AF}^D = 110,$ $\bar{S}_{ij}^k = S_{CD}^E = 160,$ checking gives the same value.
6	R = route 52.
7	$L_S = \min (3, 1, -) = 1.$
8	R_{86} is created with points D-C-E-D, loads = 1, and link E-D unloaded, calculation of S_{ED}^k gives 0, loads on $R_{52} = 3 - 1 = 2.$
9	$T_{EC} = 1 - 1 = 0, T_{DE} = 0 - 1 = -1, T_{CD} = 3 - 1 = 2,$ route 59 is eliminated as the loads are reduced to 0.
10	$S_{CD}^k = 0,$ go to Step 5.
5	Due to the many recalculation of savings and the space

TABLE 15 (CONTINUED)

<u>STEP</u>	<u>ACTION</u>
5	required, only the final maximum savings, the one used, will be noted. $\bar{S}_{ij}^k = S_{GH}^D = 130.$
6	$R = \text{route } 68.$
7	$L_S = \min (1, -, 3) = 1.$
8	R_{68} is updated to H-G-D-H, 1 load, empty link G-H, S_{GD}^k is calculated to be $S_{GD}^k = 0.$
9	$T_{DG} = 0 - 1 = -1, T_{HD} = 3 - 1 = 2, T_{GH} = 1 - 1 = 0,$ load of R_{72} are reduced to 2.
10	$S_{GH}^k = 0$ as $T_{GH} = 0$, return to Step 5.
5	$\bar{S}_{ij}^k = S_{GI}^B = 120.$
6	$R = \text{route } 69.$
7	$L_S = \min (4, -, 3) = 3.$
8	Create R_{87} with points I-G-B-I, loads = 3, and unloaded link G-B, calculate $S_{GB}^k = S_{GB}^D = 40$, reduce load of R_{69} by 3 to 1.
9	$T_{BG} = 0 - 3 = -3, T_{IB} = 3 - 3 = 0, T_{GI} = 4 - 3 = 1,$ R_{75} is eliminated due to a reduction to 0 loads.
10	$S_{GI}^k = S_{GI}^F = 120$, go to Step 5.
5	$\bar{S}_{ij}^k = S_{GI}^F = 120.$

TABLE 15 (CONTINUED)

<u>STEP</u>	<u>ACTION</u>
6	$R = \text{route } 49.$
7	$L_S = \min (1, 2, -) = 1.$
8	Update R_{69} to have points I-G-F-I, loads = 1, unloaded link G-F, calculate $S_{GF}^k = S_{GF}^D = 75.$
9	$T_{IF} = 2 - 1 = 1, T_{FG} = 0 - 1 = -1, T_{GI} = 1 - 1 = 0,$ R_{76} 's loads are reduced to 1.
10	$S_{GI}^k = 0,$ go to Step 5.
5	$S_{ij}^k = S_{AF}^D = 120.$
6	$R = \text{route } 45.$
7	$L_S = \min (5, -, 5) = 5.$
8	Point D is added to R_{45} so F-A-D-F, loads = 5, and link A-D is unloaded. Calculate $S_{AD}^k = S_{AD}^E = 55.$
9	$T_{DA} = 0 - 5 = -5, T_{FD} = 5 - 5 = 0, T_{AF} = 5 - 5 = 0,$ loads on R_{64} are reduced from 5 to 0 so the route is eliminated.
10	$S_{AF}^k = 0,$ go to Step 5.
5	$S_{ij}^k = S_{EI}^I = 90.$
6	$R = \text{route } 63.$
7	$L_S = \min (1, 1, -) = 1.$
8	R_{63} is adjusted to have points I-E-J-I, load = 1, and

TABLE 15 (CONTINUED)

<u>STEP</u>	<u>ACTION</u>
8	J-I is an empty link. Calculate S_{JI}^k which equals 0.
9	$T_{JE} = 1 - 1 = 0$, $T_{IJ} = 0 - 1 = -1$, $T_{EI} = 1 - 1 = 0$, eliminate R_{79} as the loads are reduced to 0.
10	$S_{EI}^k = 0$, go to Step 5.
5	$S_{IJ}^k = S_{BF}^G = 75$.
6	R = route 50.
7	$L_S = \min(3, 3, -) = 3$.
8	R_{50} is changed to have points F-B-G-F, an unloaded link G-F and 3 loads. Calculate $S_{GF}^k = 0$.
9	$T_{GB} = 3 - 3 = 0$, $T_{FG} = 0 - 3 = -3$, $T_{BF} = 3 - 3 = 0$, R_{65} is eliminated as its loads are reduced to 0.
10	$S_{BF}^k = 0$, return to Step 5.
5	$S_{IJ}^k = S_{AD}^E = 55$.
6	R = route 45.
7	$L_S = \min(5, 1, -) = 1$.
8	Create R_{88} with points F-A-E-D-F, 1 load and E-D as an unloaded link. Calculate $S_{ED}^k = 0$ and reduce the loads of R_{45} to 4.
9	$T_{EA} = 1 - 1 = 0$, $T_{DE} = 0 - 1 = -1$, $T_{AD} = 0 - 1 = -1$, R_{58} is eliminated as the number of loads is 0.

TABLE 15 (CONTINUED)

<u>STEP</u>	<u>ACTION</u>
10	$S_{AD}^k = 0$, go to Step 5.
5	$\bar{S}_{ij}^k = S_{CI}^F = 55$.
6	$R = \text{route } 55$.
7	$L_S = \min (1, -, 1) = 1$.
8	R_{55} is changed to have points I-C-F-I, unloaded link C-F and 1 load. Calculate $S_{CF}^k = 0$.
9	$T_{FC} = 0 - 1 = -1$, $T_{IF} = 1 - 1 = 0$, $T_{CI} = 1 - 1 = 0$, route R_{76} is eliminated.
10	$S_{CI}^k = 0$ as $T_{CI} = 0$. Return to Step 5.
5	$\bar{S}_{ij}^k = S_{GB}^D = 40$.
6	$R = \text{route } 87$.
7	$L_S = \min (3, -, 2) = 2$.
8	Establish R_{89} with points I-G-D-B-I, loads = 2, empty link G-D. Calculate $S_{GD}^k = 0$ and reduce R_{87} 's loads to 1.
9	$T_{DG} = 0 - 2 = -2$, $T_{BD} = 2 - 2 = 0$, $T_{GB} = 0 - 2 = -2$, eliminate R_{48} .
10	$S_{GB}^k = S_{GB}^C = 35$, go to Step 5.
5	$\bar{S}_{ij}^k = S_{GB}^C = 35$.
6	$R = \text{route } 87$.

TABLE 15 (CONTINUED)

<u>STEP</u>	<u>ACTION</u>
7	$L_S = \min (1, 1, -) = 1.$
8	Update R_{87} to I-G-C-B-I, loads = 1, C-B unloaded link and calculate $S_{CB}^k = 0.$
9	$T_{CG} = 1 - 1 = 0, T_{BC} = 0 - 1 = -1, T_{GB} = -2 - 2 = -3,$ eliminate R_{54} as no loads now exist on the route.
10	$S_{GB}^k = 0,$ go to Step 5.
5	$\bar{S}_{ij}^k = S_{EA}^k = 0$ for all i and j , so go to Step 11.
11	The list of routes are the solution routes. This list is given as Table 16.

TABLE 16
10 POINT EXAMPLE PROBLEM
FINAL ROUTES

ROUTE NUMBER	ROUTE POINTS	LOADS	DISTANCE	UNLOADED LINK
1	B-A-B	1	170	0
2	C-A-C	2	220	0
3	D-A-D	2	330	0
4	E-A-E	4	160	0
5	F-A-F	2	440	0
6	G-A-G	1	310	0
7	H-A-H	2	470	0
8	I-A-I	2	310	0
9	J-A-J	3	430	0
10	C-B-C	3	390	0
11	D-B-D	1	160	0
12	E-B-E	2	300	0
13	F-B-F	3	270	0
14	G-B-G	2	140	0
15	H-B-H	4	300	0
16	I-B-I	2	260	0
17	J-B-J	2	500	0
18	D-C-D	2	540	0
19	E-C-E	1	160	0
20	F-C-F	5	580	0
21	G-C-G	2	320	0
22	H-C-H	2	480	0
23	I-C-I	3	310	0
24	J-C-J	2	240	0
25	E-D-E	2	380	0
26	F-D-F	2	110	0
27	G-D-G	2	220	0
28	H-D-H	1	320	0
29	I-D-I	5	340	0
30	J-D-J	1	580	0
31	G-E-G	2	160	0
32	H-E-H	2	320	0
33	I-E-I	1	150	0
34	G-F-G	4	260	0
35	H-F-H	2	210	0
36	I-F-I	3	380	0
37	J-F-J	2	620	0
38	H-G-H	4	160	0

TABLE 16

10 POINT EXAMPLE PROBLEM
FINAL ROUTES (CONTINUED)

ROUTE NUMBER	ROUTE POINTS	LOADS	DISTANCE	UNLOADED LINK
39	I-G-I	3	120	0
40	J-G-J	5	360	0
41	I-H-I	2	210	0
42	J-H-J	4	450	0
43	J-I-J	3	240	0
44	B-A-B	1	170	A-B
45	F-A-D-F	4	440	A-D
46	G-A-J-G	2	550	0
47	H-A-D-H	1	560	A-D
49	E-B-I-E	1	355	0
50	F-B-G-F	3	335	G-F
52	D-C-D	2	540	C-D
53	F-C-F	1	580	C-F
55	I-C-F-I	1	635	C-F
56	G-D-E-G	2	380	0
57	J-D-E-J	2	615	0
61	F-E-F	2	420	E-F
62	H-E-I-H	1	340	E-I
63	I-E-J-I	1	330	J-I
67	F-G-F	2	260	G-F
68	H-G-D-H	1	350	G-D
69	I-G-F-I	1	380	G-F
71	D-H-D	2	320	H-D
72	F-H-F	1	210	H-F
73	I-H-F-I	1	400	H-F
81	I-J-I	1	240	J-I
82	J-D-H-J	2	675	0
83	D-B-J-D	2	620	0
84	H-A-I-H	2	495	0
85	E-B-C-E	1	425	0
86	D-C-E-D	1	540	E-D
87	J-G-C-B-J	1	545	C-B
88	F-A-E-D-F	1	545	E-D
89	J-G-D-B-J	2	380	G-D

2.4 Variations on the Initial Algorithm

Two modifications were made to the initial algorithm in hopes of improving the solutions. The rules for choosing between ties of equal savings and for selecting the route to update when several are possible were not changed. It was felt that these rules were specific to the situation and can be adapted to fit management's decisions.

The first variation was to eliminate step one of the algorithm resulting in not pairing the loads. This required computing a one link as well as the two link savings. The one link savings was determined by combining routes $i-j-i$, where $j-i$ was unloaded, with routes $j-i-j$ where again $i-j$ was unloaded. This savings formed a route $i-j-i$ or $j-i-j$ where both links were loaded. Thus for each $T_{ij} > 0$ two savings can now be calculated, the first is a one link savings $S_{ij1} = \alpha D_{ji}$ where $\alpha = +1$ if $T_{ji} > 0$ and $\alpha = -1$ if $T_{ji} \leq 0$ and the second is a two link savings $S_{ij2}^k = \max_{k \in p} [\alpha D_{ki} + \alpha D_{jk} + D_{ij}]$ which is the same as before.

A second variation was to add a three link savings which resulted from adding two points or three new links to a route. This three link savings was determined by calculating the savings for all combinations of two points between i and j and selecting the maximum amount as the savings S_{ij3}^{kh} . The three link saving was computed as follows: $S_{ij3}^{kj} = \max_{k, h \in p} [\alpha D_{ki} + \alpha D_{hk} + \alpha D_{jh} + D_{ij}]$ where $\alpha = +1$ if $T_{ki} > 0$ or $\alpha = -1$ if $T_{ki} \leq 0$.

It would be possible to try many variations by trying any combination of one, two, or three link savings. Preliminary studies indicated that the

initial pairing with two link savings, the one and two link savings, and the one, two, and three link savings were the best combinations. Variations to the algorithm were developed for the latter two combinations. These two algorithms follow. Notation is the same as that in section 2.3

One and Two Link Savings Algorithm

The steps of this variation to the original algorithm are listed below:

1. Create the backhaul matrix by forming the transpose of the load matrix where the elements $T_{ij} = L_{ji}$ for all i and j .
2. Create a set of two link routes $\{R\}$. A route $j-i-j$ is formed for each $T_{ij} > 0$. Each route R has a number of loads corresponding to the number of backhauls T_{ij} . As a consequence each route has an empty backhaul $i-j$.
3. The savings are now calculated. Two savings are computed for each $T_{ij} > 0$. These are the one link savings $S_{ij1} = \alpha D_{ji}$, where $\alpha = +1$ or -1 if $T_{ji} > 0$ or $T_{ji} \leq 0$ respectively and the two link savings $S_{ij2}^k = \max_{k \in p} [\alpha D_{ki} + \alpha D_{jk} + D_{ij}]$ where $\alpha = +1$ or -1 if T_{ki} or T_{jk} are >0 or ≤ 0 respectively.
4. Find the maximum savings \bar{S}'_{ij} of both the one and two link savings. Use an arbitrary rule to break ties between savings. If all $S'_{ij} \leq 0$ go to Step 10, otherwise

continue. Check the value of \bar{S}_{ij} to see if it is correct. As stated before this value may be incorrect due to the elimination of previous backhauls as a result of forming new routes. The reason this is done at this point is to eliminate the necessity of recalculating all the savings each time a new route is formed. If there is a correction in the value of \bar{S}_{ij} this step is repeated.

5. The savings \bar{S}_{ij} only applies to routes which have an empty backhaul on link $i-j$. If \bar{S}_{ij} is a one link saving, the two link route $j-i-j$ is found. If this two link route has been eliminated or expanded, set $S_{ijl} = 0$ and return to Step 4. In the case where \bar{S}_{ij} is a two link savings, this savings applies to all routes having an empty backhaul on link $i-j$. This savings may apply to several routes. If this happens, arbitrarily select one of the routes. In either the one or the two link savings situations, the route picked may have several loads.
6. The number of loads used in establishing a new route is determined by $L_s = \min [\text{number of empty backhaul loads on link } ij \text{ of route } R; T_{ki} > 0; T_{jk} > 0]$ for the two link case and $L_s = T_{ij}$ for the one link case. The

"empty backhaul loads on link $i-j$ of route R "
occur as a result of a previously formed route.

7. Form a new route R' . For the one link savings case, R' will consist of links $j-i$ and $i-j$, both with L_s loads. If \bar{S}_{ij} is a two link savings, R' will have link $i-j$ replaced by links $i-k$ and $k-j$. This route will also have L_s loads. In both cases, the new route with L_s loads may not replace all of the loads of the old route. When this occurs the old route remains with L_s less loads. When using two link savings if T_{ki} or $T_{jk} \leq 0$, empty backhauls will be created as a result of forming the new route. When this happens, the savings for the empty link $i-k$ or $k-j$ must be calculated since it is possible to improve the route by eliminating the backhaul.
8. Reduce the number of loads of the initial two link routes $i-k-i$ and $k-j-k$ by L_s if \bar{S}_{ij} is a two link savings. For the one link savings the loads of two link route $i-j-i$ are reduced by L_s . If the loads on the initial two link routes are reduced to zero, eliminate the route. Update the backhaul matrix by reducing elements T_{ki} , T_{jk} , and T_{ij} by L_s for

the two link savings but reduce only elements T_{ij} and T_{ji} by L_s if the linking results from a one link savings.

9. As a result of the above steps, the savings for link $i-j$ should be recomputed. If $T_{ij} \neq 0$ check both S_{ij1} and S_{ij2}^k , however if $T_{ij} = 0$ set both S_{ij1} and $S_{ij2}^k = 0$. Return to Step 4.
10. If no more one or two link savings are positive, no further improvements can be made. The set of $\{R\}$ are then the solution to the problem.

This algorithm has been used in the experimentation in section 2.5.

It is referred to as the one and two link savings algorithm.

One, Two, and Three Link Savings Algorithm

The listing below is the procedure for this extension of the original algorithm.

1. Create the backhaul matrix by forming the transpose of the load matrix where the elements $T_{ij} = L_{ji}$ for all i and j .
2. Create a set of two link routes $\{R\}$. A route $j-i-j$ is formed for each $T_{ij} > 0$. Each R has a number of loads corresponding to the number of backhauls T_{ij} . As a consequence, each route

- has an empty backhaul i-j.
3. The savings are now calculated. Three savings are calculated for each $T_{ij} > 0$. These are the one link savings $S_{ij1} = \alpha D_{ji}$ where $\alpha = +1$ if $T_{ji} > 0$ or -1 if $T_{ji} \leq 0$, the two link savings $\bar{S}_{ij2}^k + \max_{k \in p} [\alpha D_{ki} + \alpha D_{jk} + D_{ij}]$ where $\alpha = +1$ or -1 if T_{ki} or T_{jk} are > 0 or ≤ 0 respectively, and the three link savings $\bar{S}_{ij3}^{kh} = \max_{k, h \in p} [\alpha D_{ki} + \alpha D_{hk} + \alpha D_{jh} + D_{ij}]$ where $\alpha = +1$ or -1 if T_{ki} , T_{hk} , or T_{jh} are > 0 or ≤ 0 .
 4. Find the maximum savings \bar{S}_{ij} of all three savings matrices. Use an arbitrary rule to break ties between savings. If $\bar{S}_{ij} \leq 0$ go to Step 10, otherwise continue. Check the value of \bar{S}_{ij} to see if it is correct. As stated before this value may be incorrect due to the elimination of previous backhauls as a result of forming new routes. Again the reason this is done at this point is to eliminate the necessity of recalculating all savings each time a new route is formed. If there is a correction in the value of \bar{S}_{ij} , this step is repeated.
 5. The savings \bar{S}_{ij} only applies to routes which include an empty backhaul link i-j. If \bar{S}_{ij} is a one link

- savings, the two link route $j-i-j$ is found. If this two link route has been eliminated or expanded, set $S_{ijl} = 0$ and return to Step 4. In the cases of two and three link savings, these savings apply to all routes having an empty backhaul on link $i-j$. In some situations this savings may apply to several routes. If this happens arbitrarily select one of the routes. Note that the route may have several loads.
6. The number of loads used in establishing a new route depends upon which savings is selected as the maximum. For the one link savings, the loads $L_s = T_{ij}$. Loads for the new route R' in the case of the two link savings is $L_s = \min [\text{number of empty backhaul loads on link } i-j \text{ of route } R; T_{ki} > 0; T_{ij} > 0; T_{jh} > 0]$. The "empty backhaul loads on link $i-j$ of route R' occur as a result of a previously formed route. T_{ij} , T_{ki} , T_{jk} , T_{hk} , and T_{jh} are the initial backhaul loads.
7. Form a new route R' . For one link savings, R' will consist of links $j-i$ and $i-j$, both with L_s loads. If \bar{S}_{ij} is a two link savings R' will have link $i-j$ replaced by links $i-k$ and $k-j$. R' in the case of

a three link savings will have link $i-j$ replaced by links $i-k$, $k-h$, and $h-j$. Both of these latter cases will have L_s loads. In all three situations the new routes with L_s loads may not replace all of the loads of the old route. When this occurs the old route remains with L_s less loads. When using two or three link saving, whenever T_{ki} , T_{jk} , T_{hk} , or $T_{jh} \leq 0$, empty backhauls will be created as a result of forming the new route.

When this happens the saving for the empty link must be calculated since it is possible to improve the route by eliminating the newly created backhaul.

8. In the cases of two or three link saving, reduce the number of loads of the original two link routes $i-k-i$ and $k-j-k$ for two link savings and $i-k-i$, $k-h-k$, and $h-j-h$ for three link savings by L_s . For the one link savings the loads on the two link route $i-j-i$ are reduced by L_s . If any of the loads on two link routes created in Step 2 are reduced to zero, the route is eliminated from further consideration. Update the backhaul matrix by reducing elements T_{ij} and T_{ji} by L_s for the one link saving, reducing elements T_{ki} , T_{jk} , and T_{ij} by L_s for the two link

savings and reducing elements T_{ki} , T_{hk} , T_{jh} , and T_{ij} by L_s for three link savings.

9. As a result of the above steps, the savings for link $i-j$ should be recomputed. If $T_{ij} \neq 0$ check S_{ij1} , S_{ij2}^k , and S_{ij3}^{kh} . However if $T_{ij} = 0$ set all three savings to zero.
10. If no more of the one, two, or three link savings are greater than zero, no further improvement can be made. The set of $\{R\}$ are then the solution to the next problem.

The above two algorithms along with the original algorithm were used to solve a set of example problems and the results were compared. The experiment and the results are presented in the next section.

2.5 Experimentation

A comparison of the three methods was made. A computer program was developed which would perform any combination of the savings techniques. This program is presented in the Appendix in Section B.

Two sets of problems were used. One consisted of 40 ten point problems. The other consisted of 20 problems with 25 points each. The first set was solved using all three techniques. The second problem set was solved using only the pairing with two link savings method and the one and two link savings method. Two attempts were made to solve a 25 point problem using the one, two, and three link savings method but both were terminated after one hour of running time.

The results were evaluated on three criteria; total system distance, execution time, and number of tours formed. No other techniques were available for a direct comparison. The results of the experiments appear as Tables 17 and 18.

In general the initial pairing with two link savings procedure required less time. For the ten point problems the average execution times in seconds for the three methods were 2.75, 14.04, and 85.42 respectively on an IBM 360/50 computer. The system distance for the one, two, and three arch savings method was better than the initial algorithm's solutions for 38 of the problems. The second technique produced better solutions than the first technique for 27 of the 40 problems. However of the 13 worse solutions, 8 were less than .2% worse. Both the second and third modifications produced

solution of fewer tours. The third method resulted in the fewest number of routes being formed.

For the 25 point problem the execution time for the third method that is the one, two, and three link savings methods was so great that no solutions were obtained after one hour of computer time. Table 18 gives the results obtained for the first and second techniques only. Better total distances were realized with the second algorithm for 15 of 20 problems. Three of the remaining problems had solutions less than .2% poorer. The number of tours were reduced with the second algorithm.

2.6 Conclusions

The third algorithm using one, two, and three link savings was ruled out due to the extremely long computation times for a practical size problem. In general it resulted in the lowest total distances for the set of ten point problems. The second algorithm gave as good or better solutions for 53 of the 60 problems.

Considering the development of a dispatching procedure, the first or the second algorithm is favored. With the second algorithm fewer tours are formed making crew scheduling easier. With this in mind the second algorithm was chosen for use in the dispatching routine in Chapter 3.

TABLE 17

SOLUTIONS TO THE TEN POINT PROBLEMS

PROB.	2 LINK SAVINGS				1 AND 2 LINK SAVINGS				1, 2, AND 3 LINK SAVINGS			
	SYSTEM DISTANCE	EXEC. TIME SEC.	NO. TOURS		SYSTEM DISTANCE	EXEC. TIME SEC.	NO. TOURS		SYSTEM DISTANCE	EXEC. TIME SEC.	NO. TOURS	
1	193,990	3.65	73		189,395	24.25	61		188,850	132.73	58	
2	2,804,495	4.73	87		2,767,965	27.60	88		2,751,755	173.96	88	
3	302,695	6.21	79		296,405	22.43	78		286,460	122.36	66	
4	58,950	1.38	47		57,905	14.27	36		58,200	75.62	29	
5	19,810	1.60	14		20,035	2.88	14		18,475	15.66	12	
6	86,514	3.80	75		87,585	29.57	69		85,790	149.10	62	
7	1,258,697	7.66	89		1,243,616	25.23	88		1,232,623	140.36	89	
8	124,657	3.66	73		127,725	19.51	72		119,512	111.46	70	
9	26,463	.91	47		25,781	12.30	36		25,822	82.31	29	
10	9,184	1.27	14		8,921	2.15	13		8,423	11.56	10	
11	112,377	3.10	72		111,887	19.35	70		111,875	118.40	63	
12	1,588,910	4.03	88		1,564,316	26.82	87		1,555,517	141.21	89	
13	165,182	3.70	77		161,655	16.92	72		156,945	121.98	67	
14	34,699	1.02	47		34,702	14.40	34		34,598	77.73	28	
15	13,300	1.25	13		13,304	2.22	13		12,813	12.99	11	
16	4,438	3.45	75		4,450	18.66	68		4,438	114.71	65	
17	60,852	4.41	88		60,504	23.08	89		59,794	118.91	89	
18	6,312	3.08	78		6,248	16.05	78		6,210	100.93	66	
19	1,274	.83	47		1,276	9.78	36		1,258	68.38	36	
20	448	1.55	14		448	2.35	14		446	12.28	12	
21	53,155	2.95	47		53,200	10.62	41		53,116	65.53	34	
22	52,925	2.20	47		52,886	11.95	41		52,835	70.73	32	
23	50,853	2.37	49		50,380	11.02	41		50,380	73.31	35	
24	50,518	1.76	49		49,578	10.73	41		47,289	67.44	34	

TABLE 17 (CONTINUED)

PROB.	2 LINK SAVINGS				1 AND 2 LINK SAVINGS				1, 2, AND 3 LINK SAVINGS			
	SYSTEM DISTANCE	EXEC. TIME SEC.	NO. TOURS		SYSTEM DISTANCE	EXEC. TIME SEC.	NO. TOURS		SYSTEM DISTANCE	EXEC. TIME SEC.	NO. TOURS	
25	45,849	2.27	48		45,873	10.90	39		45,413	70.38	35	
26	138,564	2.89	72		138,809	15.80	67		136,676	103.14	64	
27	1,860,396	4.03	88		1,824,764	17.18	89		1,803,395	113.13	86	
28	204,050	3.78	75		201,323	15.18	72		200,774	98.46	64	
29	40,254	1.03	46		39,817	11.42	33		39,716	70.85	28	
30	14,440	1.59	12		14,487	2.68	13		13,928	16.87	8	
31	19,440	2.99	46		19,250	10.15	39		19,150	62.41	35	
32	21,405	2.04	49		21,235	11.00	38		20,955	75.64	36	
33	19,720	2.40	51		19,535	11.65	41		19,480	74.58	36	
34	19,330	1.80	49		18,845	10.60	46		18,570	70.66	36	
35	18,845	2.13	49		18,635	10.96	42		18,515	67.47	32	
36	53,430	3.00	72		53,580	15.05	66		53,275	111.90	63	
37	727,835	3.75	88		718,875	16.60	88		710,225	115.23	89	
38	68,455	3.33	72		76,710	14.13	71		76,275	99.90	70	
39	14,810	1.37	45		14,730	11.51	33		14,710	72.38	28	
40	5,525	1.17	13		5,525	2.51	11		5,490	13.95	11	
AVG.		2.75				14.04				85.42		

TABLE 18

SOLUTIONS TO THE 25 POINT PROBLEMS

PROB.	2 LINK SAVINGS			1 AND 2 LINK SAVINGS			1, 2, AND 3 LINK SAVINGS		
	SYSTEM DISTANCE	EXEC. TIME SEC.	NO. TOURS	SYSTEM DISTANCE	EXEC. TIME SEC.	NO. TOURS	SYSTEM DISTANCE	EXEC. TIME SEC.	NO. TOURS
1	372,612	68.33	311	366,510	455.64	261	NO SOLUTIONS OBTAINED DUE TO EXCESSIVE EXECUTION TIMES.		
2	363,557	76.63	259	355,199	357.13	234			
3	247,796	72.27	204	249,391	265.52	182			
4	136,544	39.70	107	136,571	112.84	97			
5	90,118	32.60	74	90,263	61.31	71			
6	201,941	73.33	311	200,802	428.06	258			
7	192,267	72.55	267	187,508	286.58	227			
8	137,774	60.50	208	135,826	259.23	181			
9	68,793	28.36	107	68,776	88.39	101			
10	47,845	30.63	75	48,444	56.29	72			
11	361,630	80.63	302	356,216	471.30	252			
12	350,184	80.35	250	349,964	461.17	220			
13	236,971	91.83	194	237,301	378.42	173			
14	128,964	59.97	97	128,805	180.36	87			
15	84,295	45.77	69	83,939	104.03	64			
16	367,558	127.41	246	365,573	446.74	219			
17	198,933	111.24	302	197,998	566.71	248			
18	187,599	82.58	255	185,305	316.66	222			
19	134,998	95.23	194	134,025	329.93	169			
20	64,820	52.80	95	65,567	175.07	89			
AVG.		69.14			290.67				

CHAPTER 3

3.1 Dispatching Routine

A problem which occurs as a part of a fleet operation is one of carrier dispatching. Carrier dispatching is defined as the assignment of carriers to routes. As routes are restructured by the completion of deliveries and new demands, new truck assignments must be created periodically. This results in a routing period or intervals for which demands are grouped and a route system formed. An example of such a period would be a day or four hours. The routing periods are subdivisions of a larger planning period, as a week or month. The nature of the system is such that some restraints apply to planning periods and others to the routing intervals. Dispatching is further complicated by such considerations as crew scheduling, restraints on the system, and multiple loads on a route.

One of the objectives of this study is to develop a dispatching routine for the line haul routing problem which include some of these features. Such a routine should contain the following:

1. Be based on a good routing technique.
2. Have the flexibility to consider in-process routes in later periods.
3. Establish routes which conform to the system constraints.
4. Form routes considering the location and the availability of carriers.

In the following discussion a dispatching routine is developed

utilizing the one and two link savings version of the algorithm. This version was selected as it was felt the savings generated outweighed the extra cost of computer time which resulted from not pairing the initial loads. The algorithm using one, two, and three link savings was ruled out due to its excessive execution time.

Only in a limited situation is it possible to incorporate in-process routes in later routing periods. Note that if a route has all of its links loaded, there is no advantage in considering it later. A route having an empty link which the carrier has already started or traveled has no possibility for improvement. This leaves the case where an empty link exists on a route and the carrier has not traveled the empty link.

The carrier's route position depends upon the link lengths, travel velocities, docking time at each facility, length of routing period, and the number of periods the carrier has been traveling the route. The problem of the carrier's route position would be eliminated if the operators periodically notified management of their location.

To include the possibility of considering in-process routes in the next routing period, several modifications were made to the computer program. For each route of a solution the carrier position at the beginning of next routing period was determined. This required the use of an average vehicle velocity. The total distance traveled was estimated using the length of a routing period, number of periods the vehicle has been traveling the route, and its average velocity. To eliminate the assignment of a carrier to an

empty link near the end of a routing period, a "maximum variation" in the length of a travel period was used. That is if the total distance less this "variation distance" was less than the route distance to the empty link, the carrier is retained at the start of the empty link and the route is considered in the next routing period. Dock times were not considered but were assumed to have been included in the vehicle velocity.

Several types of restrictions could be considered such as the maximum length of a route, the maximum time allowed for traveling the route, and the maximum number of links on a route. Restrictions on individual routes would be required to insure the completion of routes in a given planning period. The method for checking restrictions was to check the new or updated route against the constraints. If none were violated the assignment was made; if some were, another route with the same empty link was picked. If all the routes violated a restraint when updated, the savings was zeroed. A sub-routine was added to the program to check route restrictions. See Section A of the Appendix for the procedure to follow in using this routine.

The availability of the carriers could limit the forming of initial two link routes. Assignment of carriers to certain links on some basis was avoided by assuming that the carrier availability was either zero or infinity. In other words, carriers could be restricted to originating from selected points. If the availability of carriers was limited, no guarantee that all the loads would be routed could be made. The above procedures were added to the computer program. An example problem was solved to illustrate the procedure.

3.2 Example Problem

In this problem the carriers were assigned to three points and routes were restricted in length. There were 15 points and a planning period of five days was used. Two routings were made each day. All of the carriers had to return to their origin by the end of the day. This limited the morning routes to 8 hours and, assuming a velocity of 30 miles per hour, to 240 miles. The afternoon routes were limited to 120 miles or 4 hours. The in-process routes from the morning were incorporated into the afternoon routing. An adjustment was made to the distances of the morning routes so that the mileage constraint would not be violated when they were considered in the afternoon routing. A carrier was not allowed to start an unloaded link if it occurred within one hour of the end of the morning routing period.

An unlimited number of carriers were available at points 6, 8, and 10. This limitation created a situation in which all the loads were not routed. Unrouted loads were considered in the next routing period.

The problem was solved for all ten routing periods. Table 20 gives the organization of the solution. The tables giving the loads used and the routes formed are given along with information about the routing system. Figure 4 illustrates the relationship of the points. Distances between the points appear as Table 19.

A total of 518 routes were created. These routes had a total distance of 73,404 miles. Of the 87 in-process routes considered in later routings, 50 were improved.

All routes observed the restriction on route length. No problems occurred in considering in-process routes. All the routes originated from one of the three points and ended at the same point. An average computation time of 41.29 seconds for each routing was required.

The routes formed were given in tables referred to by Table 20. Points on a route were given as $8 + 2 + 3 + 10 - 8$, where $+$ designates a loaded link and $-$ an empty link. Also, the route length and number of loads were given.

FIGURE 4

DISTANCE NETWORK FOR EXAMPLE PROBLEM

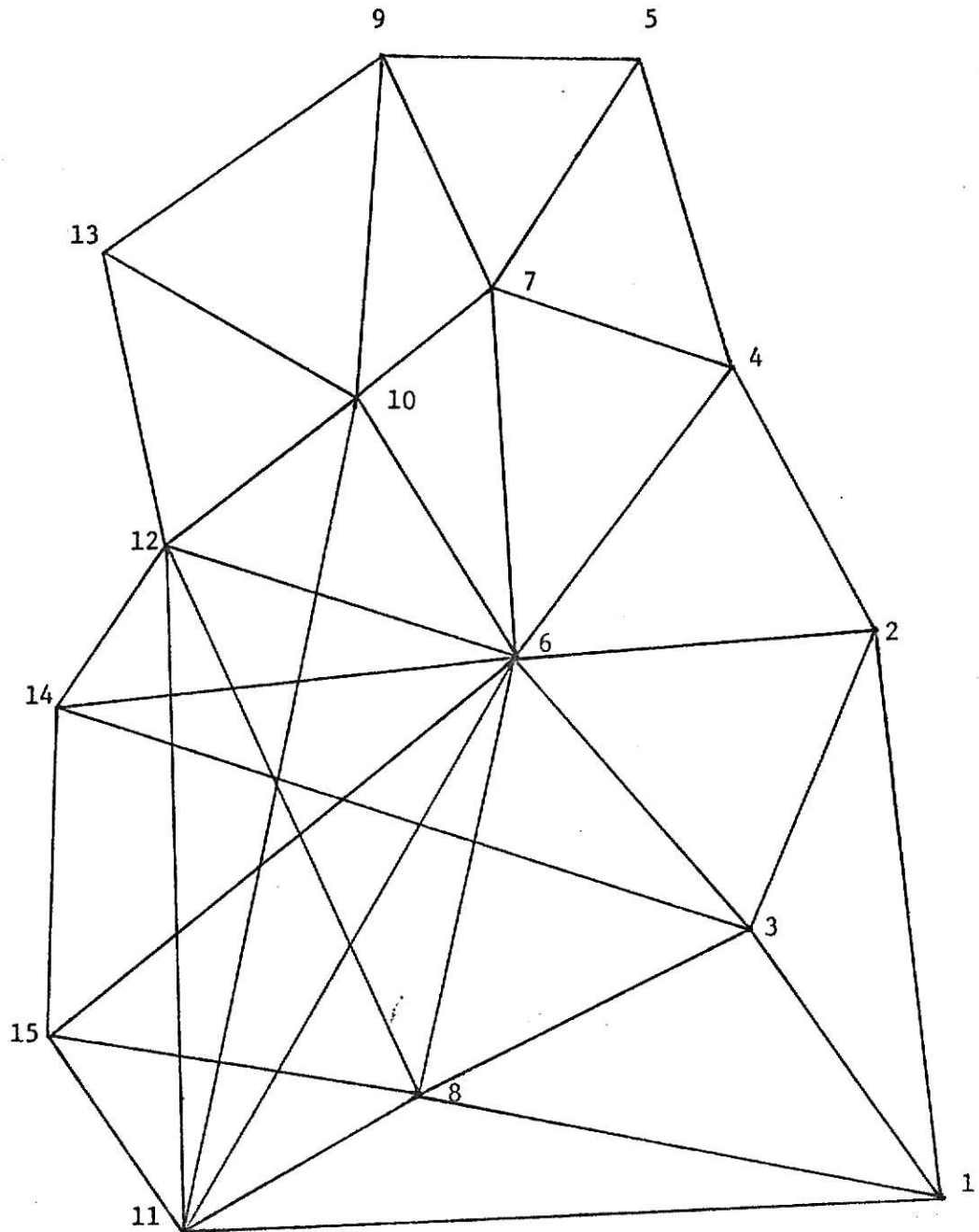


TABLE 19

DISTANCE MATRIX FOR EXAMPLE PROBLEM

POINT

P
O
I
N
T

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	0	31	18	47	65	38	58	29	72	55	42	58	71	58	50
2	31	0	18	16	34	20	30	39	44	37	54	40	53	46	53
3	18	18	0	34	52	20	40	21	54	37	36	40	53	40	42
4	47	16	34	0	18	20	14	45	28	23	56	36	39	46	53
5	65	34	52	18	0	35	15	60	14	24	71	37	32	48	66
6	38	20	20	20	35	0	20	25	34	17	36	20	33	26	33
7	58	30	40	14	15	20	0	45	14	9	56	22	25	33	51
8	29	39	21	45	60	25	45	0	59	42	15	33	50	39	21
9	72	44	54	28	14	34	14	59	0	19	65	32	18	43	61
10	55	37	37	23	24	17	9	42	19	0	46	13	16	24	42
11	42	54	36	56	71	36	56	15	65	46	0	37	54	31	13
12	58	40	40	36	37	20	22	33	32	13	37	0	17	11	29
13	71	53	53	39	32	33	25	50	18	16	54	17	0	28	46
14	58	46	40	46	48	26	33	39	43	24	31	11	28	0	18
15	50	53	42	53	66	33	51	21	61	42	13	29	46	18	0

TABLE 20

SOLUTION OF EXAMPLE PROBLEM

<u>Routing Period</u>	<u>Load Matrix Table</u>	<u>Number Of Routes</u>	<u>Distance Of Routes</u>	<u>Number Of Inprocess Routes</u>	<u>Number Of Routes Changed</u>	<u>Solution Table</u>	<u>Solution Time In Seconds</u>
1	21	44	7267	15	0	22	43.51
2	23	13	1419	0	8	24	23.65
3	25	52	6999	0	0	26	47.20
4	27	45	5396	0	0	28	41.53
5	29	66	10108	24	0	30	32.45
6	31	47	5988	0	15	32	31.76
7	33	67	9040	27	0	34	64.38
8	35	54	6950	0	13	36	28.33
9	37	74	11435	21	0	38	54.88
10	39	56	6802	0	14	40	45.18
Totals		518	73404	87	50		412.87
Averages		51.8	7340.4	21.75	12.5		41.29

TABLE 21

LOAD MATRIX FOR PERIOD 1 OF EXAMPLE PROBLEM

TO POINT

FROM
POINT

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	0	1	2	1	0	0	1	5	1	1	0	1	0	1	1
2	1	0	1	5	2	0	1	0	0	1	0	1	0	2	1
3	1	1	0	1	2	0	1	0	1	0	5	0	1	3	0
4	1	3	1	0	0	1	0	0	0	1	0	1	5	0	1
5	1	3	2	1	0	1	0	0	2	1	0	0	1	1	0
6	3	3	1	1	0	0	0	1	0	0	1	1	2	0	0
7	1	3	5	1	0	1	0	1	0	0	0	0	1	0	1
8	2	1	1	0	1	1	1	0	1	0	1	0	2	3	1
9	1	0	2	1	0	1	0	1	0	1	0	1	2	3	4
10	3	1	0	2	1	0	2	1	0	0	1	0	2	5	1
11	4	1	0	0	1	0	2	0	1	0	0	1	2	0	0
12	1	1	0	2	1	1	0	1	0	1	1	0	1	2	1
13	2	1	0	1	0	0	0	0	1	0	1	2	0	1	2
14	1	1	1	1	0	1	2	0	0	1	0	0	1	0	1
15	1	2	1	5	0	0	1	0	1	1	0	0	1	1	0

TABLE 22
SOLUTION ROUTES FOR PERIOD 1 OF EXAMPLE PROBLEM

Route Periods	Points	Length	Number Loads
1 and 2	$6 + 1 + 3 + 13 - 6 + 3 + 6$	222	1
1	$8 + 1 + 7 + 8$	132	1
1	$10 + 1 + 12 + 10$	126	1
1	$6 + 2 + 15 + 2 - 6 + 2 + 6$	238	1
1	$8 + 2 + 12 + 8$	112	1
1	$10 + 2 + 5 + 10$	95	1
1	$6 + 3 + 9 + 6$	108	1
1 and 2	$8 + 3 + 11 + 1 + (10) + 8$	196	1
1	$6 + 4 + 12 + 6$	76	1
1	$10 + 4 + 13 - 10 + 12 + 4 + 10$	170	1
1	$8 + 5 + 1 + 8$	154	1
1	$10 + 5 + 9 + 10$	57	1
1	$8 + 6 + 8$	50	1
1	$8 + 7 + 1 + 8$	132	1
1 and 2	$10 + 7 + 3 + 5 + 14 + (5) + 10$	221	1
1	$10 + 8 + 14 - 10$	105	1
1	$8 + 9 + 1 + 8$	160	1
1	$6 + 11 + 5 + 6$	142	1
1	$8 + 11 + 1 - 8$	86	1
1	$10 + 11 + 2 + 10$	137	1
1	$6 + 12 + 1 - 6$	116	1
1	$6 + 13 + 4 + 15 + 2 + 4 - 6$	214	1
1	$8 + 13 + 1 + 8$	150	2
1 and 2	$10 + 13 + 15 + 7 + 3 + 14 + (12) - 10$	217	1
1	$10 + 14 - 10$	48	1
1 and 2	$8 + 15 + 3 + 11 + 1 + (15) + 8$	212	1
1	$10 + 15 + 1 + 10$	147	1
1	$8 + 1 + 9 + 8$	160	1
1	$10 + 1 + 15 + 10$	147	1
1	$10 + 1 + 14 + 10$	137	1
1	$6 + 1 + 4 + 6$	105	1
1	$10 + 14 + 4 + 10$	93	1
1	$6 + 2 + 14 + 6$	92	1
1	$6 + 2 + 7 + 6$	70	1
1 and 2	$8 + 14 + 1 + 3 + 11 + 1 - 8$	222	1
1 and 2	$6 + 13 + 2 + 5 + 3 + (4) + 6$	226	1
1 and 2	$8 + 14 + 3 + 11 + 7 - 8$	216	1

TABLE 22 (CONTINUED)

Route Period	Points	Length	Number Loads
1 and 2	$6 + 1 + 2 + 3 + 5 + 3 - 6$	211	1
1 and 2	$10 + 14 + 7 + 3 + 14 + (9) - 10$	199	1
1 and 2	$10 + 14 + 13 + 15 + 9 + 12 - 10$	204	1
1 and 2	$10 + 13 + 11 + 7 + 3 + 14 + (12) - 10$	230	1
1 and 2	$10 + 4 + 13 + 14 - 10 + 12 + 4 + 10$	206	1
1 and 2	$10 + 7 + 13 + 9 + 14 + (9) - 10$	157	1
1 and 2	$10 + 14 + 7 + 15 + 4 + 13 - 10$	216	1

TABLE 23

LOAD MATRIX FOR PERIOD 2 OF EXAMPLE PROBLEM

TO POINT

FROM POINT		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	1	0	1	0	0	2	0	0	1	0	1	0	0	0	0	1
	2	1	0	1	4	0	0	3	0	0	0	1	0	0	1	0
	3	1	1	0	1	3	0	0	0	0	1	1	0	3	0	0
	4	2	3	2	0	2	1	0	0	1	0	0	0	1	0	0
	5	1	3	0	3	0	0	1	0	1	1	2	0	2	1	1
	6	0	1	3	0	1	0	0	0	0	1	0	1	0	1	0
	7	1	0	0	1	2	0	0	0	1	0	2	3	0	1	0
	8	0	0	1	0	0	1	0	0	0	0	1	0	2	0	0
	9	3	2	1	0	0	0	0	0	0	0	1	0	1	2	1
	10	0	1	0	0	0	1	0	3	0	0	0	2	0	1	0
	11	0	0	0	3	0	1	0	0	1	0	0	1	1	0	0
	12	0	2	0	3	1	0	0	0	3	0	1	0	2	1	4
	13	2	1	2	3	0	4	5	0	1	0	2	3	0	0	1
	14	0	0	0	0	1	0	0	1	2	0	0	3	0	0	1
	15	0	0	0	4	0	3	0	2	0	1	5	0	3	1	0

TABLE 24

SOLUTION ROUTES FOR PERIOD 2 OF EXAMPLE PROBLEM

Route Periods	Points	Length	Number Loads
2	6 + 2 + 11 + 6	110	1
2	10 + 2 + 14 + 12 - 10	107	1
2	6 + 3 + 13 + 6	106	3
2	8 + 3 + 10 + 8	100	1
2	8 + 6 + 14 + 8	90	1
2	10 + 6 + 5 - 10	76	1
2	10 + 8 + 13 - 10	108	1
2	6 + 10 + 12 + 13 + 6	80	1
2	8 + 11 + 4 - 8	116	1
2	6 + 12 + 15 + 6	82	1
2	8 + 13 - 8	100	1
2	10 + 14 - 10	48	1
2	10 + 12 + 15 + 10	84	1

TABLE 25

LOAD MATRIX FOR PERIOD 3 EXAMPLE

TO POINT

FROM
POINT

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	0	1	0	0	2	1	2	1	3	2	0	1	3	2	0
2	1	0	1	4	2	0	3	2	4	1	0	0	1	0	1
3	1	1	0	0	3	3	0	4	0	3	1	2	0	0	1
4	2	3	2	0	2	2	0	0	1	0	0	2	1	0	0
5	1	3	0	3	0	5	1	5	1	5	2	1	2	1	1
6	1	0	3	0	1	0	1	0	3	2	5	0	1	2	1
7	1	0	0	1	2	0	0	2	1	3	2	3	0	1	0
8	1	4	2	1	0	3	1	0	2	2	0	1	1	2	3
9	3	2	1	1	2	5	0	1	0	1	1	0	1	1	1
10	3	2	1	0	1	3	2	3	5	0	4	2	0	1	1
11	0	0	0	2	0	0	1	3	1	0	0	1	1	3	0
12	0	2	1	3	1	1	0	4	3	2	1	0	1	1	2
13	2	1	2	3	0	3	5	0	1	3	2	3	0	0	1
14	1	0	0	1	1	2	0	2	3	0	0	0	2	0	1
15	1	2	0	4	0	2	0	1	0	1	5	0	3	1	0

TABLE 26

SOLUTION ROUTES FOR PERIOD 3 OF EXAMPLE PROBLEM

Route Periods	Points	Length	Number Loads
3	6 + 1 + 9 + 6	144	1
3	8 + 1 + 9 + 8	160	1
3	10 + 1 + 5 + 10	144	2
3	8 + 2 + 7 + 8	114	1
3	10 + 2 + 13 + 10	106	1
3	6 + 3 + 10 + 6	74	2
3	8 + 3 + 5 + 8	133	2
3	10 + 3 + 5 + 10	113	1
3	8 + 4 + 12 + 8	114	1
3	6 + 5 + 15 + 6	134	1
3	10 + 5 + 12 + 10	74	1
3	8 + 6 + 10 + 8	84	2
3	10 + 6 + 1 + 7 + 1 + 7 + 10	238	1
3	8 + 7 + 14 + 8	117	1
3	10 + 7 + 11 + 10	111	2
3	6 + 9 + 14 + 6	103	1
3	8 + 9 + 2 + 8	142	1
3	10 + 9 + 6 + 10 + 14 + 9 + 10	209	1
3	8 + 10 + 11 + 8	103	1
3	6 + 11 + 14 + 6	93	1
3	10 + 11 + 14 + 13 + 10	121	1
3	8 + 12 + 3 + 8	94	1
3	10 + 12 + 4 + 13 + 7 + 10 + 12 + 10	194	1
3	6 + 13 + 3 + 6	106	1
3	8 + 13 + 2 + 8	142	1
3	6 + 14 + 9 + 6	103	2
3	8 + 14 + 15 + 8	78	1
3	6 + 15 + 13 + 6	112	1
3	8 + 15 + 2 + 3 + 8	113	1
3	10 + 15 + 1 + 10	147	1
3	8 + 9 + 1 + 8	160	1
3	8 + 14 + 5 + 8	147	1
3	10 + 1 + 9 + 10	146	1
3	10 + 9 + 1 + 10	146	1
3	6 + 9 + 1 + 6	144	1
3	6 + 11 + 9 + 6	135	1
3	8 + 2 + 5 + 8	133	2

TABLE 26 (CONTINUED)

Route Periods	Points	Length	Number Loads
3	10 + 2 + 15 + 10	132	1
3	6 + 9 + 15 + 6	128	1
3	6 + 11 + 13 + 6	123	1
3	8 + 2 + 10 + 8	118	1
3	6 + 11 + 4 + 6	112	2
3	10 + 11 + 7 + 10	111	1
3	10 + 9 + 3 + 10	110	1
3	8 + 10 + 14 + 8	105	1
3	10 + 11 + 12 + 10	96	1
3	8 + 6 + 7 + 8	90	1
3	6 + 3 + 12 + 6	80	1
3	10 + 12 + 5 + 10	74	1
3	10 + 9 + 5 + 10	57	1
3	10 + 9 + 13 + 10	53	1
3	8 + 15 + 11 + 8	49	2

TABLE 27

LOAD MATRIX FOR PERIOD 4 OF EXAMPLE PROBLEM

TO POINT

F
R
O
M

P
O
I
N
T

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	0	1	0	0	3	0	1	2	0	1	0	1	3	2	0
2	1	0	1	4	0	1	2	0	4	5	0	0	0	1	1
3	1	1	0	3	0	2	0	2	2	3	1	1	3	0	1
4	2	3	2	0	2	0	5	1	1	2	0	3	0	0	0
5	1	3	0	0	0	5	1	1	1	5	2	0	2	1	0
6	0	0	3	5	0	0	3	10	0	1	3	2	1	1	1
7	0	0	0	1	2	0	0	0	1	3	0	3	0	0	2
8	3	2	1	1	0	1	2	0	1	0	4	5	0	1	1
9	0	1	0	1	1	0	0	4	0	5	0	2	0	4	0
10	5	1	0	3	3	3	5	0	2	0	4	0	5	3	1
11	0	0	0	0	5	0	1	1	0	1	0	0	1	1	0
12	1	2	0	1	0	0	0	3	3	3	1	0	1	1	2
13	2	0	1	3	0	1	4	0	1	10	2	3	0	3	1
14	1	0	0	1	0	3	0	0	2	1	0	0	1	0	0
15	0	1	0	4	0	1	0	3	0	1	3	0	2	1	0

TABLE 28

SOLUTION ROUTES FOR PERIOD 4 OF EXAMPLE PROBLEM

Route Periods	Points	Length	Number Loads
4	8 + 1 + 12 + 8	120	1
4	10 + 1 - 10	110	4
4	10 + 2 + 10	74	1
4	6 + 3 + 9 - 6	108	1
4	6 + 4 + 12 - 6	76	1
4	10 + 4 + 2 - 10	76	2
4	10 + 5 + 10	48	1
4	8 + 6 + 12 + 8	78	1
4	6 + 7 + 15 + 14 + 6	115	1
4	8 + 7 + 9 + 8	118	1
4	10 + 7 + 12 + 10	44	3
4	6 + 8 + 3 - 6	66	1
4	8 + 9 + 8	118	1
4	10 + 9 + 14 + 10	86	1
4	6 + 10 + 5 + 6	76	1
4	6 + 11 - 6	72	2
4	8 + 11 - 8	30	3
4	10 + 11 - 10	92	4
4	6 + 12 + 14 + 6	57	1
4	6 + 13 + 15 + 6	112	1
4	10 + 13 - 10	32	5
4	6 + 14 + 4 - 6	92	1
4	10 + 14 - 10	48	3
4	6 + 15 + 13 + 6	112	1
4	8 + 15 + 4 + 8	119	1
4	10 + 15 - 10	84	1
4	8 + 7 + 5 + 8	120	1
4	10 + 7 + 15 + 10	102	1
4	10 + 9 + 2 + 10	100	1
4	10 + 6 + 11 + 10	99	1
4	6 + 8 + 14 + 6	90	1
4	6 + 8 + 2 + 6	84	1
4	6 + 3 + 10 + 6	74	2
4	6 + 4 + 3 + 6	74	2
4	6 + 4 + 5 + 6	73	2
4	6 + 7 + 5 + 6	70	1

TABLE 28 (CONTINUED)

Route Periods	Points	Length	Number Loads
4	10 + 1 + 10	110	1
4	6 + 8 + 4 - 6	90	1
4	10 + 7 + 4 + 10	46	1
4	6 + 8 + 2 - 6	84	1
4	6 + 7 + 10 + 5 + 6	88	1
4	6 + 8 + 12 - 6	78	5
4	8 + 1 + 8	58	2
4	10 + 4 + 10	46	1
4	8 + 11 + 8	30	1

TABLE 29

LOAD MATRIX FOR PERIOD 5 OF EXAMPLE PROBLEM

TO POINT

FROM
POINT

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	0	1	0	0	3	0	1	0	5	0	2	1	3	2	1
2	1	0	1	4	0	0	2	0	4	1	1	1	2	1	1
3	1	1	0	3	0	1	0	2	1	1	1	1	3	2	1
4	2	1	0	0	0	0	5	0	1	0	1	2	0	1	1
5	1	3	0	0	0	1	0	0	1	4	2	0	2	1	0
6	4	6	5	3	3	0	0	0	1	0	0	3	0	2	1
7	0	0	1	0	3	0	0	3	0	2	2	0	0	0	1
8	4	3	4	2	1	0	2	0	0	0	0	3	0	1	1
9	1	0	0	1	1	0	0	1	0	5	0	2	0	3	0
10	5	0	7	1	0	1	3	2	2	0	0	2	3	0	1
11	0	1	3	0	5	0	1	0	1	0	0	0	1	1	3
12	1	2	0	1	1	0	0	1	3	0	1	0	1	2	2
13	2	2	1	3	0	1	4	0	1	5	2	3	0	3	0
14	1	1	0	1	0	1	1	1	2	0	0	0	1	0	1
15	0	1	0	3	0	1	0	3	0	0	3	0	1	1	0

TABLE 30
SOLUTION ROUTES FOR PERIOD 5 OF EXAMPLE PROBLEM

Route Periods	Points	Length	Number Loads
5	6 + 1 + 15 + 6	121	1
5 and 6	8 + 1 + 11 + 3 + (9) + 8	220	1
5	10 + 1 + 5 + 10	144	1
5 and 6	6 + 2 + 9 + 14 + (3) + 6	167	1
5	8 + 2 + 3 + 8	78	1
5	6 + 3 - 6	40	1
5	8 + 3 - 8	42	1
5 and 6	10 + 3 + 14 + 9 + (11) + 10	231	1
5	6 + 4 + 7 - 6	54	1
5 and 6	8 + 4 + 11 + 2 + (1) - 8	215	1
5	10 + 4 + 12 - 10	72	1
5 and 6	6 + 5 + 11 + 5 - 6	212	1
5	8 + 5 + 14 + 8	147	1
5	8 + 7 + 3 + 8	106	1
5	10 + 7 - 10	18	2
5 and 6	10 + 8 + 12 + 9 + (15) - 10	210	1
5 and 6	6 + 9 + 1 + 13 + (12) + 6	214	1
5	10 + 9 + 12 - 10	64	1
5	6 + 12 + 4 + 7 - 6	90	1
5 and 6	8 + 12 + 1 + 11 + 3 - 8	190	1
5 and 6	10 + 12 + 9 + 4 + 9 - 10	120	1
5	10 + 13 + 7 + 10	50	2
5	6 + 14 + 4 + 7 - 6	106	1
5	8 + 14 + 7 + 8	117	1
5 and 6	6 + 15 + 4 + 7 + (1) + (12) - 6	236	1
5 and 6	8 + 15 + 4 + 2 + 9 + (15) + 8	216	1
5	10 + 15 + 2 + 10	132	1
5	8 + 1 + 9 + 8	160	1
5	10 + 1 + 9 + 10	146	4
5	6 + 1 + 13 + 6	142	1
5	6 + 1 + 5 + 6	138	1
5	8 + 1 + 7 + 8	132	1
5	6 + 1 + 14 + 6	122	1
5	8 + 1 + 12 + 8	120	1
5	8 + 4 + 15 + 8	119	1
5	8 + 7 + 15 + 8	117	1

TABLE 30 (CONTINUED)

Route Periods	Points	Length	Number Loads
5	8 + 2 + 7 + 8	114	1
5	8 + 2 + 15 + 8	113	1
5	10 + 3 + 9 + 10	110	1
5	10 + 3 + 13 + 10	106	3
5	10 + 13 + 3 + 10	106	1
5	6 + 5 + 10 + 6	76	1
5	10 + 12 + 5 + 10	74	1
5	6 + 5 + 11 + 3 + 6	162	1
5 and 6	10 + 3 + 12 + 9 + (15) - 10	212	1
5	10 + 9 + 5 + 10	57	1
5	10 + 3 + 14 + 13 + 10	121	1
5 and 6	6 + 14 + 2 + 9 + 14 + (4) - 6	225	1
5	10 + 8 + 12 + 13 + 10	108	1
5 and 6	8 + 3 + 11 + 5 + 1 - 8	222	1
5	8 + 3 + 15 - 8	84	1
5 and 6	6 + 2 + 12 + 15 + 4 + (14) - 6	214	1
5	6 + 2 + 4 - 6 + 1 + 2 + 6	211	1
5 and 6	6 + 2 + 14 + 1 + 13 + (12) + 6	232	1
5 and 6	6 + 2 + 13 + 1 + 14 - 6	228	1
5 and 6	6 + 4 + 14 + 15 + 14 - 6 + 14 + 6	214	1
5	6 + 12 + 11 - 6 + 9 + 12 + 6	209	1
5 and 6	6 + 2 + 11 + 5 + (3) + 6	180	1
5 and 6	6 + 4 + 12 + 15 + 13 + 2 + 4 - 6	220	1
5	6 + 3 + 4 - 6	74	1
5 and 6	6 + 12 + 2 + 7 + 11 - 6	182	1
5 and 6	6 + 3 + 4 + 1 + (5) + 6	201	1
5 and 6	6 + 3 + 4 + 1 + 5 - 6	201	1
5 and 6	10 + 7 + 11 + 5 + 13 - 10	184	1
5	6 + 3 + 2 + 4 - 6	74	1
5	8 + 3 + 1 - 8	68	1

TABLE 31

LOAD MATRIX FOR PERIOD 6 OF EXAMPLE PROBLEM

TO POINT

FROM
POINT

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	0	0	1	0	3	0	1	0	0	1	0	4	0	0	1
2	1	0	0	1	1	1	0	0	0	1	1	1	0	1	0
3	0	0	0	0	3	3	0	0	5	1	1	0	1	1	1
4	1	3	1	0	2	0	1	4	0	0	5	0	2	4	1
5	0	3	1	1	0	1	2	0	1	0	2	0	1	3	3
6	1	1	3	5	1	0	2	4	3	3	0	1	1	1	2
7	3	0	0	2	3	1	0	0	2	1	0	0	0	1	1
8	1	1	0	3	0	2	4	0	1	3	5	1	1	2	1
9	0	0	0	0	5	0	3	1	0	1	1	0	0	0	4
10	1	3	1	5	2	1	5	6	0	0	3	2	6	1	1
11	0	1	2	0	1	0	1	1	1	3	0	0	1	1	3
12	2	0	1	1	0	3	0	0	4	1	1	0	1	2	0
13	0	1	0	3	0	1	2	0	1	0	2	3	0	3	1
14	4	0	1	1	1	0	1	1	4	0	0	1	0	0	0
15	2	0	0	4	2	0	1	1	0	0	3	0	0	1	0

TABLE 32

SOLUTION ROUTES FOR PERIOD 6 OF EXAMPLE PROBLEM

Route Periods	Points	Length	Number Loads
6	6 + 1 + 7 - 6	116	1
6	8 + 1 - 8	58	1
6	10 + 1 + 10	110	1
6	6 + 2 + 12 + 6	80	1
6	10 + 2 + 5 - 10	95	1
6	6 + 3 + 9 - 6	108	2
6	10 + 3 + 11 + 10	119	1
6	6 + 4 + 11 - 6	112	5
6	10 + 4 + 14 + 7 - 10	111	1
6	6 + 5 - 6	70	1
6	10 + 5 - 10	48	2
6	8 + 6 + 10 + 8	84	2
6	6 + 7 + 1 - 6	116	2
6	8 + 7 - 10 + 8	96	2
6	10 + 7 - 10	18	2
6	6 + 8 + 4 - 6	90	3
6	6 + 9 - 6	68	3
6	8 + 9 - 8	118	1
6	6 + 10 + 8 + 2 - 6	118	1
6	8 + 10 + 11 + 8	103	1
6	8 + 11 - 8	30	5
6	10 + 11 - 10	92	2
6	6 + 12 + 3 + 6	80	1
6	8 + 12 + 4 + 8	114	1
6	10 + 12 - 10	26	2
6	6 + 13 + 2 + 6	106	1
6	10 + 13 + 4 - 10	78	3
6	6 + 14 - 6	52	1
6	8 + 14 - 8	78	2
6	10 + 14 + 5 - 10	96	1
6	6 + 15 + 4 - 6	106	1
6	8 + 15 + 4 + 8	119	1
6	10 + 15 - 10	84	1
6	8 + 7 + 14 + 8	117	1
6	10 + 13 + 11 + 10	116	1
6	8 + 10 + 4 + 8	110	2

TABLE 32 (CONTINUED)

Route Periods	Points	Length	Number Loads
6	6 + 8 + 13 + 6	108	1
6	6 + 15 + 7 + 6	104	1
6	8 + 7 + 10 + 8	96	1
6	6 + 3 + 10 + 6	74	1
6	10 + 13 + 9 + 10	53	1
6	10 + 13 + 12 + 10	46	1
6	10 + 2 - 10	74	1
6	10 + 2 + 14 - 10	107	1
6	10 + 4 + 13 + 7 - 10	96	2
6	10 + 7 + 15 - 10	102	1
6	10 + 7 + 9 + 7 - 10	46	2

TABLE 33
LOAD MATRIX FOR PERIOD 7 OF EXAMPLE PROBLEM

TO POINT

FROM
POINT

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	0	0	1	0	2	0	1	3	0	2	0	3	2	0	1
2	1	0	0	1	0	2	0	4	3	0	1	0	1	1	0
3	0	0	0	0	3	0	0	1	2	0	0	0	1	1	1
4	1	3	1	0	2	1	1	0	0	5	1	3	0	2	1
5	0	3	0	1	0	0	2	0	1	0	2	0	1	3	3
6	1	2	3	0	2	0	1	1	5	0	1	1	0	0	1
7	1	0	0	2	3	0	0	1	0	2	4	0	0	3	0
8	3	0	2	5	2	4	1	0	3	3	2	4	0	2	3
9	1	1	0	0	5	1	3	0	0	0	4	0	0	1	1
10	3	3	2	1	0	5	6	2	4	0	3	3	2	1	2
11	0	1	2	0	1	0	1	0	1	0	0	0	1	1	3
12	2	1	1	1	0	0	0	1	4	1	1	0	1	2	0
13	1	4	0	0	0	0	0	0	0	0	1	0	0	3	1
14	4	1	2	1	0	0	0	0	4	0	0	1	0	0	0
15	2	1	0	2	2	0	4	0	0	0	3	0	0	1	0

TABLE 34

SOLUTION ROUTES FOR PERIOD 7 OF EXAMPLE PROBLEM

Route Periods	Points	Length	Number Loads
7 and 8	$6 + 1 + 12 + (4) + (14) - 6$	204	1
7	$8 + 1 + 3 - 8 + 1 + 8$	126	1
7 and 8	$10 + 1 + 13 + 2 - 10$	216	1
7	$6 + 2 + 11 - 8 + 6$	114	1
7	$10 + 2 + 9 - 10$	100	1
7 and 8	$6 + 3 + 5 + 15 + 4 - 6$	211	1
7 and 8	$8 + 3 + 14 + 1 + (13) - 8$	240	1
7 and 8	$10 + 3 + 13 + (11) + 10$	190	1
7 and 8	$8 + 4 + 15 + 11 + (9) - 8$	235	1
7	$10 + 4 + 7 + 10$	46	1
7 and 8	$6 + 5 + 11 + 2 + (14) - 6$	232	1
7	$8 + 5 + 2 + 8$	133	2
7	$8 + 6 - 8$	50	1
7	$10 + 6 - 10$	34	2
7	$6 + 7 + 11 - 8 + 6$	116	1
7	$8 + 7 + 1 + 8$	132	1
7	$10 + 7 - 10$	18	1
7 and 8	$6 + 9 + 14 + 1 + 5 - 6$	235	1
7 and 8	$8 + 9 + 11 - 8$	139	1
7	$10 + 9 - 10$	38	1
7	$6 + 11 + 9 + 6$	135	1
7	$8 + 11 - 8$	30	1
7 and 8	$10 + 11 + 15 + 7 - 10 + 11 + 10$	227	1
7	$6 + 12 + 2 - 6$	80	1
7	$8 + 12 + 3 - 8$	94	1
7	$10 + 12 + 9 - 10$	64	2
7 and 8	$10 + 13 + 2 + 9 + (15) + 10$	216	2
7 and 8	$8 + 14 + 1 + (9) + 8$	228	1
7	$10 + 14 + 4 + 10$	93	1
7	$6 + 15 + 2 + 6$	106	1
7 and 8	$8 + 15 + 7 + 11 - 8$	143	1
7	$10 + 15 + 1 + 10$	147	2
7	$8 + 9 + 1 + 8$	160	1
7	$8 + 9 + 2 + 8$	142	1
7	$8 + 1 + 7 + 8$	132	1
7	$10 + 1 + 8 + 10$	126	1

TABLE 34 (CONTINUED)

Route Periods	Points	Length	Number Loads
7	10 + 1 + 12 + 10	126	1
7	8 + 14 + 2 + 8	124	1
7	8 + 1 + 12 + 8	120	1
7	10 + 11 + 7 + 10	111	1
7	8 + 4 + 10 + 8	110	2
7	10 + 3 + 8 + 10	100	1
7 and 8	10 + 11 + (5) + 11 + 10	234	1
7	8 + 4 + 6 + 8	90	1
7	6 + 5 + 2 + 6	89	1
7 and 8	8 + 4 + 11 + (5) + 8	236	1
7	10 + 2 + 4 + 10	76	1
7 and 8	10 + 2 + 13 + 14 + (7) + 10	160	1
7	10 + 12 + 4 + 10	72	1
7 and 8	6 + 9 + 11 - 8 + 6	139	1
7 and 8	6 + 9 + 15 + 4 + 14 - 6	220	1
7 and 8	8 + 12 + 1 + (5) + 8	216	1
7 and 8	8 + 12 + 1 + 15 - 8	162	1
7	8 + 12 + 11 - 8	85	1
7 and 8	6 + 9 + 11 + 3 + 5 - 6	222	2
7	8 + 3 + 15 - 8	84	1
7 and 8	6 + 2 + 14 + 1 + 5 - 6	224	1
7 and 8	6 + 3 + 9 + 5 + 7 + (12) + (7) - 10 + 6	217	1
7	10 + 6 - 8 + 10	84	1
7	8 + 15 + 11 - 8 + 15 + 8	181	2
7	10 + 9 + 7 - 10	42	3
7 and 8	10 + 7 + 11 + 15 + 7 - 10	138	1
7 and 8	10 + 7 + 11 + 15 + 7 + (12) + (13) - (10) + (7) + 10	202	1
7	10 + 7 + 14 + 12 + 9 - 10 + 5 + 14 + 10	209	1
7	10 + 7 + 14 + 3 - 10 + 5 + 14 + 10	224	2
7 and 8	6 + 3 + 9 + 5 + 13 + 2 - 10 + 6	227	1
7 and 8	8 + 11 + 15 + 7 + 4 + 1 - 8	169	1

TABLE 35

LOAD MATRIX FOR PERIOD 8 OF EXAMPLE PROBLEM

TO POINT

FROM POINT		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	1	0	0	1	2	3	1	0	0	1	0	1	2	1	0	0
	2	1	0	0	1	0	3	0	1	0	2	0	0	1	1	1
	3	0	3	0	2	4	1	0	0	1	0	1	0	3	2	0
	4	0	3	1	0	2	1	0	0	1	0	0	3	1	1	0
	5	0	1	0	1	0	0	1	3	1	0	4	0	5	0	1
	6	3	2	1	0	1	0	2	3	5	4	6	7	0	1	0
	7	1	2	0	1	3	0	0	0	1	2	4	5	0	0	1
	8	0	1	3	5	2	4	1	0	2	3	4	0	3	1	1
	9	0	0	1	0	0	3	1	1	0	2	0	1	2	0	4
	10	1	1	5	3	2	4	3	1	1	0	3	2	1	1	1
	11	2	1	0	1	1	0	0	4	3	2	0	1	0	0	3
	12	0	1	4	5	0	0	6	0	1	0	0	0	1	2	0
	13	4	0	1	0	1	0	1	0	2	1	1	0	0	2	1
	14	0	0	1	2	0	0	5	0	1	1	4	0	0	0	0
	15	0	1	0	0	3	2	1	0	0	2	0	0	1	1	0

TABLE 36
SOLUTION ROUTES FOR PERIOD 8 OF EXAMPLE PROBLEM

Route Periods	Points	Length	Number Loads
8	6 + 1 + 12 - 6	116	2
8	6 + 2 + 13 - 6	106	1
8	8 + 2 + 6 + 8	84	1
8	6 + 3 + 2 + 6	58	1
8	8 + 3 - 8	42	3
8	10 + 3 + 2 + 10	92	1
8	8 + 4 - 8	90	5
8	10 + 4 + 12 + 4 - 10	118	2
8	8 + 5 - 8	120	2
8	10 + 5 - 10	48	2
8	8 + 6 + 11 - 8	76	1
8	10 + 6 + 9 - 10	70	1
8	6 + 7 + 15 + 6	104	1
8	8 + 7 + 11 + 8	116	1
8	10 + 7 + 11 - 10	111	3
8	10 + 8 + 9 + 10	120	1
8	6 + 9 - 6	68	4
8	10 + 9 - 10	38	1
8	6 + 10 + 11 - 6	99	2
8	6 + 11 + 4 - 6	112	1
8	8 + 11 - 8	30	1
8	10 + 11 - 10	92	3
8	6 + 12 + 4 - 6	76	1
8	10 + 12 + 7 + 12 - 10	70	1
8	8 + 13 - 8	100	3
8	10 + 13 - 10	32	1
8	6 + 14 + 9 + 6	103	1
8	8 + 14 + 11 + 8	85	1
8	10 + 14 - 10	48	1
8	8 + 15 - 8	42	1
8	10 + 15 - 10	84	1
8	8 + 6 + 5 + 8	120	1
8	10 + 3 + 11 + 10	119	1
8	6 + 8 + 9 + 6	118	1
8	6 + 7 + 1 + 6	116	1

TABLE 36 (CONTINUED)

Route Periods	Points	Length	Number Loads
8	10 + 3 + 9 + 10	110	1
8	8 + 11 + 2 + 8	108	1
8	6 + 2 + 15 + 6	106	1
8	10 + 3 + 13 + 10	106	1
8	6 + 1 + 4 + 6	105	1
8	10 + 3 + 14 + 10	101	1
8	10 + 12 + 2 + 10	90	1
8	6 + 12 + 9 + 6	86	1
8	10 + 6 + 8 + 10	84	1
8	6 + 12 + 3 + 6	80	1
8	8 + 6 + 11 + 8	76	2
8	6 + 10 + 2 + 6	74	1
8	10 + 4 + 12 + 7 + 10	90	1
8	6 + 12 + 3 - 6	80	3
8	6 + 12 + 4 + 3 - 6	110	1
8	10 + 6 + 11 - 8 + 10	110	2
8	6 + 10 + 1 - 6	110	1
8	8 + 11 + 1 - 8	86	1
8	8 + 11 + 1 + 11 - 8	114	1

TABLE 37

LOAD MATRIX FOR PERIOD 9 OF EXAMPLE PROBLEM

TO POINT

FROM
POINT

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	0	0	1	1	2	0	3	2	1	0	0	1	2	0	3
2	1	0	0	1	0	0	2	0	1	1	0	0	2	3	0
3	0	1	0	2	4	4	1	2	0	0	2	0	2	1	0
4	1	3	0	0	2	1	0	2	1	0	0	3	1	0	1
5	0	1	0	1	0	0	1	0	1	0	4	0	5	0	1
6	2	5	1	4	3	0	2	1	2	0	2	3	4	1	2
7	0	2	0	1	3	1	0	0	1	2	4	5	0	0	0
8	1	3	2	4	2	0	1	0	3	1	4	3	3	2	1
9	0	0	1	3	0	0	1	0	0	4	0	0	2	0	2
10	3	2	1	4	5	1	1	0	2	0	2	3	3	3	3
11	1	1	1	0	3	3	0	0	2	0	0	1	4	0	3
12	0	0	0	4	0	0	3	1	1	0	2	0	1	2	0
13	4	0	1	0	1	0	1	0	2	0	2	2	0	2	1
14	0	0	1	2	0	0	4	0	0	0	3	0	0	0	0
15	0	1	0	1	3	0	1	0	1	5	0	0	1	1	0

TABLE 38
SOLUTION ROUTES FOR PERIOD 9 OF EXAMPLE PROBLEM

Route Periods	Points	Length	Number Loads
9	$6 + 1 + 3 + 6$	76	1
9	$8 + 1 + 12 + 8$	120	1
9	$10 + 1 + 15 + 10$	147	3
9	$6 + 2 - 6$	40	1
9 and 10	$8 + 2 + 14 + 4 + 2 + (14) + 8$	232	1
9 and 10	$8 + 2 + 14 + 4 + 2 - 8$	186	1
9 and 10	$10 + 2 + 13 + (1) - 10$	216	1
9 and 10	$6 + 3 + 13 + 14 + 7 + 12 + 14 - 6$	193	1
9	$8 + 3 + 11 - 8$	72	2
9	$10 + 3 + 7 + 10$	86	1
9	$6 + 4 - 6$	40	2
9	$8 + 4 + 12 - 8$	114	2
9	$10 + 4 + 5 - 10$	65	1
9	$6 + 5 + 11 + 6$	142	3
9 and 10	$8 + 5 + 2 + (9) + 8$	197	1
9	$10 + 5 + 13 - 10$	72	1
9	$10 + 6 + 8 + 10$	84	1
9 and 10	$6 + 7 + 11 + 13 + 14 + (3) - 6$	218	1
9 and 10	$8 + 7 + 11 + (4) + (2) - 8$	212	1
9	$10 + 7 - 10$	18	1
9	$6 + 9 + 4 - 6$	82	1
9	$8 + 9 + 4 + 8$	132	2
9	$10 + 9 + 13 - 10$	53	1
9 and 10	$6 + 11 + 5 + 4 - 6$	145	1
9	$8 + 11 + 13 - 8$	119	2
9	$10 + 11 + 9 + 10$	130	1
9	$6 + 12 + 4 - 6$	76	3
9	$8 + 12 - 8$	66	1
9	$10 + 12 + 7 - 10$	44	2
9 and 10	$6 + 13 + 11 + 5 + (3) - 6$	230	1
9	$8 + 13 + 3 + 8$	124	1
9	$10 + 13 - 10$	32	2
9	$6 + 14 + 3 + 6$	86	1
9	$8 + 14 + 11 - 8$	85	2
9	$10 + 14 + 7 - 10$	66	2

TABLE 38 (CONTINUED)

Route Periods	Points	Length	Number Loads
9	$6 + 15 + 2 - 6$	106	1
9 and 10	$8 + 15 + 5 + (1) + 8$	181	1
9 and 10	$10 + 15 + 5 + (8) + 10$	210	1
9	$8 + 13 + 1 + 8$	150	2
9	$10 + 11 + 2 + 10$	137	1
9	$8 + 9 + 3 + 8$	134	1
9	$10 + 5 + 15 + 10$	132	1
9	$10 + 9 + 15 + 10$	122	1
9	$10 + 15 + 9 + 10$	122	1
9	$6 + 1 + 7 + 6$	116	1
9 and 10	$8 + 5 + 11 + (4) - 8$	232	1
9	$6 + 15 + 4 + 6$	106	1
9	$10 + 15 + 7 + 10$	102	1
9	$10 + 2 + 9 + 10$	100	1
9	$6 + 11 + 3 + 6$	92	1
9 and 10	$8 + 4 + 15 + (4) + 8$	196	1
9	$10 + 4 + 9 + 10$	70	1
9 and 10	$6 + 13 + 1 + 7 + 2 - 6$	212	1
9 and 10	$8 + 4 + 1 + 13 + 15 - 8$	230	1
9 and 10	$6 + 9 + 15 + 5 + 7 + 12 - 6$	218	1
9	$8 + 12 + 11 - 8$	85	2
9 and 10	$8 + 2 + 14 + 11 + (4) - 8$	217	1
9	$10 + 4 + 12 + 13 - 10$	92	1
9	$10 + 4 + 13 - 10$	78	1
9 and 10	$6 + 2 + 13 + 7 + 11 + (15) - 6$	200	1
9	$6 + 7 + 12 - 6 + 9 + 7 + 6$	197	1
9 and 10	$6 + 13 + 11 + 5 + 13 - 6$	223	1
9	$6 + 2 + 7 + 12 + 14 - 6$	109	1
9 and 10	$8 + 11 + 1 + 13 + 12 + (9) - 8$	236	1
9	$10 + 12 + 4 - 10$	72	1
9	$10 + 5 + 9 - 10 + 5 + 10$	121	1
9 and 10	$6 + 2 + 1 + 9 + (8) + 3 + 6$	223	1
9	$6 + 2 + 7 + 4 - 6 + 7 + 6$	128	1
9	$10 + 13 + 12 + 9 + 13 - 10$	99	1
9	$8 + 11 + 12 - 8$	85	1
9 and 10	$6 + 13 + 1 + 7 + 2 + 4 - 6$	228	1
9	$6 + 4 + 2 - 6$	56	1
9	$10 + 5 + 13 + 9 - 10$	93	2
9	$10 + 14 + 7 + 9 - 10$	90	1
9	$6 + 4 + 5 - 6$	73	1

TABLE 39

LOAD MATRIX FOR PERIOD 10 OF EXAMPLE PROBLEM

TO POINT

F
R
O
M

P
O
I
N
T

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	0	0	0	1	2	0	0	1	0	0	4	0	0	3	3
2	2	0	0	1	1	1	0	2	1	2	0	3	0	4	2
3	0	1	0	2	4	0	0	0	2	0	0	1	1	1	0
4	0	1	1	0	0	2	3	1	0	1	0	1	0	0	3
5	2	0	4	1	0	0	1	1	1	0	0	1	1	1	0
6	1	0	1	3	4	0	5	1	1	0	2	3	2	1	1
7	0	2	4	0	3	0	0	0	5	5	0	1	0	1	0
8	1	2	5	2	2	3	1	0	1	1	5	4	5	1	0
9	0	2	0	4	2	0	3	4	0	1	1	2	1	0	0
10	0	1	3	3	3	0	1	2	2	0	3	3	4	2	1
11	3	0	0	5	0	1	0	0	0	4	0	0	1	0	3
12	0	0	1	0	4	0	0	5	3	3	0	0	0	2	0
13	2	0	1	0	4	2	1	0	0	3	1	1	0	0	0
14	0	2	3	0	0	2	0	1	1	0	0	1	0	0	1
15	2	0	1	1	0	1	1	1	0	3	0	0	1	1	0

TABLE 40
SOLUTION ROUTES FOR PERIOD 10 OF EXAMPLE PROBLEM

Route Periods	Points	Length	Number Loads
10	6 + 1 - 6	76	1
10	8 + 1 + 15 - 8	100	1
10	8 + 2 + 8	78	2
10	10 + 2 - 10	74	1
10	6 + 3 + 14 + 6	86	1
10	8 + 3 - 8	42	4
10	10 + 3 + 12 + 10	90	1
10	6 + 4 + 7 + 14 - 6	93	1
10	8 + 4 + 12 + 8	114	1
10	10 + 4 + 7 + 10	46	1
10	6 + 5 + 3 - 6	107	2
10	8 + 5 - 8	120	2
10	10 + 5 + 3 - 10	113	1
10	8 + 6 + 12 + 8	78	2
10	6 + 7 + 2 + 12 - 6	110	1
10	8 + 7 + 9 + 8	118	1
10	10 + 7 + 12 + 5 - 10	92	1
10	6 + 8 + 13 + 6	108	1
10	10 + 8 + 2 + 10	118	1
10	10 + 9 - 10	38	2
10	6 + 11 - 6	72	1
10	8 + 11 + 4 - 8	116	2
10	10 + 11 + 10	92	3
10	6 + 12 + 3 - 6	80	1
10	8 + 12 - 8	66	2
10	10 + 12 + 5 - 10	74	2
10	6 + 13 - 6	66	2
10	8 + 13 - 8	100	4
10	10 + 13 + 10	32	1
10	6 + 14 + 3 - 6	86	1
10	8 + 14 - 8	78	1
10	10 + 14 + 15 + 10	84	1
10	6 + 15 + 13 + 6	112	1
10	10 + 15 + 7 + 10	102	1
10	10 + 8 + 9 + 10	120	1
10	8 + 4 + 15 + 8	119	1

TABLE 40 (CONTINUED)

Route Periods	Points	Length	Number Loads
10	10 + 4 + 15 + 10	118	2
10	8 + 6 + 9 + 8	118	1
10	10 + 13 + 11 + 10	116	1
10	6 + 5 + 14 + 6	109	1
10	10 + 14 + 2 + 10	107	1
10	10 + 3 + 13 + 10	106	1
10	10 + 3 + 4 + 10	94	1
10	10 + 5 + 12 + 10	74	1
10	6 + 5 + 4 + 6	73	1
10	10 + 5 + 13 + 10	72	1
10	10 + 13 + 7 + 10	50	1
10	10 + 13 + 12 + 10	46	1
10	6 + 7 + 3 - 6	80	3
10	6 + 7 + 3 + 4 + 6	114	1
10	6 + 11 + 6	72	1
10	6 + 4 + 3 - 6	74	1
10	8 + 12 + 8	66	2
10	8 + 11 + 1 - 8	86	3
10	10 + 12 + 5 + 7 + 10	74	1
10	6 + 4 + 7 + 2 - 6	84	1

3.3 Conclusion

The results obtained show that the routine works. There exist many refinements and improvements which could be made. Some of these would depend upon the actual problem being analyzed.

Two other problems of this same type were solved. In both five routings were made. One consisted of 25 points where the routes were unrestricted. The other had 10 points and routes were restricted in length. Neither had any restrictions on the availability of carriers. In both some routes were considered in three later routings. The average execution times were 152.79 and 15.28 seconds. Of the in-process routes 32 out of 68 were improved for the 25 point case and 19 out of 39 for the 10 point situation.

It would be possible to repeatedly run the program with a set of demands to form a simulation. In this way the selection of central points could be evaluated as well as different size carriers, and other system restraints.

REFERENCES

1. Automobile Manufacturers Association, "1970 Motor Truck Facts," (1970).
2. Balinski, M. L., and Quendt, R. E., "On an Integer Program for a Delivery Problem," Operations Research, Vol. 12, 300-304 (1964).
3. Bellman, R., "On a Routing Problem," Quarterly Journal of Applied Mathematics, Vol. 16, No. 1, 87-90 (1958).
4. Bellmore, M., and Nemhauser, G. L., "The Traveling Salesman Problem: A Survey," Operations Research, Vol. 16, 538-558 (1968).
5. Braun, W., "A Computerized Simulation Approach to the Solution of the Carrier Dispatching Problem," Master's Thesis Kansas State University (1967).
6. Central Engineering Institute of Research, "Techniques for Truck and Driver Scheduling," unpublished report by C. E. I. R. for the American Trucking Association (1969).
7. Clarke, G., and Wright, J. W., "Scheduling of Vehicles from a Central Depot to a Number of Delivery Points," Operations Research, Vol. 12, No. 4, 568-581 (1964).
8. Cochran, H., "Optimization of a Carrier Routing Problem," Master's Thesis Kansas State University (1967).
9. Dantzig, G. B., Fulkerson, D. R., and Johnson, S. M., "On a Linear Programming Combinatorial Approach to the Traveling-Salesman Problem," Operations Research, Vol. 7, 58-66 (1959).
10. Dantzig, G. B., Fulkerson, D. R., and Johnson, S. M., "Solution of a Large-Scale Traveling-Salesman Problem," Operations Research, Vol. 2, 393-410 (1954).
11. Dantzig, G. B., and Ramser, J. H., "The Truck Dispatching Problem," Management Science, Vol. 6, No. 1, 80-91 (1959).

12. Datz, I. M., "Simulated Shipping," Datamation, Vol. 12, No. 2, 61-63 (1966).
13. Flood, M. M., "The Traveling-Salesman Problem," Operations Research, Vol. 4, 61-75 (1956).
14. Gaskell, T. J., "Bases for Vehicle Fleet Scheduling," Operation Research Quarterly, Vol. 18, No. 3, 281-295 (1967).
15. Hausman, W. H., and Gilmour, P., "A Multi-Period Truck Delivery Problem," Transportation Research, Vol. 1, 349-357 (1967).
16. Hering, R., "Evaluation of Some Heuristic Look Ahead Rules for Multiple Terminal Delivery Problems," Master's Thesis Kansas State University, (1970).
17. Karg, R. L., and Thompson, G. L., "A Heuristic Approach to Solving Traveling Salesman Problems," Management Science, Vol. 10, 225-248 (1964).
18. Knight, K. W., and Hofer, J. P., "Vehicle Scheduling with Timed and Connected Calls: A Case Study," Operations Research Quarterly, Vol. 19, No. 3, 229-310 (1968).
19. Little, J. D. C., Murty, K. G., Sweeney, D. W., and Karel, C., "An Algorithm for the Traveling Salesman Problem," Operations Research, Vol. 11, 972-989 (1963).
20. Miller, C. E., Tucker, A. H., and Zemlin, R. A., "Integer Programming Formulation of Traveling Salesman Problems," Association for Computing Machinery, Vol. 7 No. 4, 326-329 (1960).
21. Nace, G. E., "Distributing Goods by VSP/360," Software Age, 8-29 (1969).
22. Nicholson, T. A. J., "A Boundary Method for Planar Traveling Salesman Problems," Operations Research Quarterly, Vol. 19, No. 4, 445-452 (1968).
23. Pierce, J. F., "Direct Search Algorithm for Truck-Dispatching Problems," Transportation Research, Vol. 3, No. 1, 1-42 (1969).

24. Tillman, F. A., "The Multiple Terminal Delivery Problem with Probabilistic Demands," Transportation Science, Vol. 3, No. 3, 192-204 (1969).
25. Tillman, F. A., and Cain, T., "An Upper Bound Algorithm for the Single and Multiterminal Delivery Problem," paper presented at the XVII TIMS International Conference, London, July 1-3, 1970.
26. Tillman, F. A., and Cochran, H., "A Heuristic Approach for Solving the Delivery Problem," Journal of Industrial Engineering, Vol. 19, No. 7, 354-358 (1968).
27. Tillman, F. A., and Herring, R., "A Study of a Look Ahead Procedure for Solving the Multiterminal Delivery Problem," to appear in Transportation Research, 1971.
28. Tyagi, M. S., "A Practical Method for Truck Dispatching Problems," Journal Operations Research Society of Japan, Vol. 10, Nos. 3 & 4, 76-92 (1968).

APPENDIX

Section A

Program User's Instructions

The Program. This is the program written for the execution of the devised algorithm with provisions for dispatching. The program permits any combination of one link savings, two link savings and three link savings to be used. Provisions are included for route restrictions, for identifying and improving inprocess routes and limiting carriers to certain points.

Inprocess routes are found by calculating the carrier's position on the route and comparing it with the location of the empty link on the route. The carrier's position is determined by using the length of the travel period, number of periods the carrier has been traveling the route, average velocity, and the variation in length of the travel period. A route is not considered if all links were loaded or the carrier had passed all the empty links. A procedure is used to enter the inprocess routes in the next time period.

Data Input. The program is designed so that several problems could be solved in sequence. Data can be divided into control cards required for the set of problems and data required by each problem. Extensive use is made of user supplied formats. This permits easy adaptation of the program to existing sets of data. The required data consists of the problem set parameters, the problem parameters, restrictions and carrier location data, demands distances between points, location and number of available carriers, and any

inprocess routes.

Use of variable formats permit the user to supply the field organization from which each type of data will be read. Formats should meet these criteria: begin in card column one; begin and end with parenthesis; be specified on one card; and be of the same type as the variable being initialized. It is assumed that all input is by means of cards. The following is an illustration of a format card.

```
cc 1 2 3 4 5 6 7 8 9 10 11 12 13 14
   ( 1 0 1 5 , 3 F 7 . 2 )
```

A subroutine RESTR was created for checking limitations on the routes. The only limitation used in this program is the maximum length of a route. Another influencing criteria for route forming is the availability of carriers at a given point. The availability of carriers at each point is considered in such a way that carriers originated and terminated only at specific points.

The algorithm was programmed using Fortran and adopted to Kansas State University's IBM 360/50 G-level compiler. Program modification would be quite simple for use of the Watfiv compiler. This entails the deletion of the unnecessary function TIME and the related statements. EBCDIC was used to punch the program. All variables, except those beginning with U and V are implicitly declared to be binary. Letters A to N begin variable which are only half words. All remaining variables are a full word in length. Read and write device parameters are variables ORD and OWT which are initialized at the first of the main routine.

The following is a description of the required data in the order in which it is read in the program.

Problem Set Parameter Cards. A format card is read followed by the problem set parameter card. Two values are supplied by this card. The number of problems to be executed and a data check indicator.

```

( 2 1 )
A B

```

A - number of problems to be executed.

B - echo check indicator, if "one"
input data is printed.

This card is read only once for each use of the program. The following data is required for each problem in the set.

Problem Parameter Cards. First a format card is read which is used to read the problem parameter card and the restrictions and carrier location card. All of these values are integers. Eight parameters are initialized by the problem parameter card.

```

( 8 2 )
C D E F G H I J
K L M N

```

C - number of points in the problem.

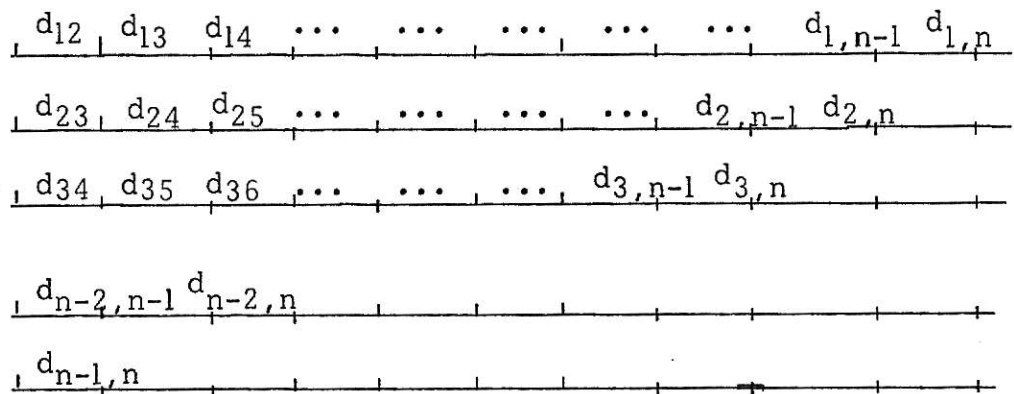
D - one link savings indicator, if "1"
one link savings are used.

E - two link savings indicator, if "1"
two link savings are used.

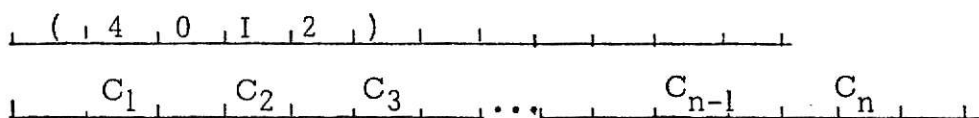
- F - three link savings indicator, if "1" three link savings are used.
- G - if "1" previous inprocess routes are read.
- H - if "1" the load matrix is not read, if "greater than 1," matrix is read and stored.
- I - if "1" the distance matrix is not read.
- J - if "1" the inprocess solution routes are punched.
- K - the maximum route length in integer units.
- L - average carrier velocity in integer units.
- M - length of travel period in integer units.
- N - units of + or - variation allowed in a travel period to allow a carrier to remain or get to a facility.

Load Demand Cards. If the value of H in the problem parameter card is a value other than "1" the load matrix is read. First an integer format is read, followed by the number of loads originating at point i and destined for j. These form a C x C matrix. Input is row by row. Each row must begin on a new card and may use as many cards as necessary. If this matrix is not read, the previously stored load matrix is used. If a series of problems use the same load matrix, the first problem should read and store the load matrix. This is done by setting H of the problem parameter card to a value "greater than 1." Note that the load matrix is altered during execution.

Distance Cards. Next the distance matrix is read if I of the problem parameter card is not "1". If the matrix is not read, the previous problems distances are used. Distances are not changed during the execution of a problem. When the distance matrix is read, an integer format is read first. The distance matrix is assumed to be symmetric so only the upper triangle is read. Distances are read row by row with each row beginning with the distances immediately to the right of the diagonal. Row i consists of n-i distances beginning with distance $d_{i,i+1}$ and ending with $d_{i,n}$. If there are n points only n-1 rows are read. Each row should begin on a new card and may use as many cards as necessary.



Carrier Availability Cards. Now an integer format card is read followed by a card with the number of carriers available for assigning at each point. These values are read as integers. They are read in order for the carriers at point 1, carriers at point 2, and on for all n points.



C_1 - number of carrier available at point i.

If previous routes are not considered this is all of the data required for each problem.

Cards to Enter Previous Routes. The data to enter previous routes will now be described. The first card is a format card to enter the remaining values. Next the number of routes to be read is entered.

```

  ( 1 6 I 5 )
  0

```

0 - number of inprocess routes
being entered.

The following three cards are read for each of the 0 routes. First the empty link and then the time period of the route are read. The second card supplies the number of loads on the route, route length, and the number of links. The third card specifies the order of and which points are on the route. The points are entered in order, ending with the originating point.

```

  P Q R
  S T U
  Pi . . . Pu

```

P - originating point of empty link.

Q - destination point of the empty
link.

R - number of travel periods which
have elapsed since the route
was assigned.

S - number of loads on the route.

T - length of the route.

U - number of links on the route.

P_i - point on the route

These cards are repeated for each route. An example of a set of input appears as Section D of this Appendix.

(2) The Output

The program output consists of a data echo check and the routes of the solution. The echo of the input prints out the original input which includes the number of points under consideration, the savings options used, maximum route length average carrier velocity, the average travel period, and the variation allowed in the running period. Printed out as a vector is the number and location of available carriers. The load and distance matrices are printed and labeled.

A print-out follows as Section C of this Appendix. The solution print-out contains the points on each route, beginning with the originating point and continuing in order. For each route the number of loads, route length, number of links, the empty length, which links are empty, the carrier's position, and the route's time period are printed. These quantities are clearly labeled and printed for each solution route.

Following the routes of the solutions are the total number of tours, total mileage, total empty mileage, the calculated load miles, execution time for the problem, and the number of inprocess routes. Last is a list of unrouted loads including the points and number of loads in each case.

The echo check is in the subroutine DATAIN at the end of the subroutine. The solution print-out procedure is located at the last of the main program. Other quantities can be printed depending upon the situation. Some additional useful output might be the number of tours completed during the time period considered. Statistics on the average route lengths and their variations can be obtained.

Program Limitations and Restrictions. As implied in the section "Data Input Procedure" there is provision provided for placing restrictions on routes formed. The only restricted applied at present is on the maximum route length. A subroutine RESTR is provided for this and additional restrictions. If the variable NSZ is set equal to "0" the route under consideration will not be picked for improvement. Some other restrictions which might be considered are the number of links a route may consist of, the maximum time for the completion of a route, and so on.

Improvements are possible for several rules which are used in the algorithm. One is the route to choose for updating when several routes contain the same empty link. Another one is the initial assignment of carriers to which routes when there is a shortage of carriers at a point. The answers to these rules most likely will be greatly influenced by the particular situation for which a solution is desired.

Code Variables. Following is a list and description of important variables used in the program. A description of the storage field is given if more than one value is stored in a field. For dimensioned variables an estimate of

the required dimensions is given. Variables beginning with letters A to N are implicitly declared binary half words. Variables beginning with letters O to T and W to Z are binary whole words, while U or V denote floating point variables.

ART(I,J)	route storage, I is route number, if J = 1 quantity stored is loads, J = 2 mileage stored, J = 3 number of links stored, J = 4 ... points on route stored, dimension $I \geq$ maximum number of expected routes, $J \geq$ maximum number of expected points on any route.
ARTLG	the maximum route length.
AVEL	average velocity between any points.
AVT(I)	number of carriers available at point I, dimension $I \geq$ maximum number of points.
AWP	average length of travel period.
BS(I,J,K)	empty links corresponding to savings SV(I,J,K), each field has three values each one digit in length, dimension $I = J \geq$ maximum number of points and $K = 3$.
DIST(I,J)	distance matrix, dimension $I = J \geq$ maximum number of points.
DVRT	the plus and minus deviation allowed in the average travel period, allows a carrier to remain at or reach a point near the end of a travel period.
ECHO	data echo check indicator, if equal one print-out of data made.
FMT(I)	variable format storage, $I = 20$.
INDEX(I,J,K)	stores the route number of the routes which have an empty link I to J, dimension $I = J \geq$ maximum number of points and $K \geq$ maximum number of loads between any I and J.
LDX(I,J)	load matrix, dimension $I = J \geq$ maximum number of points.

LDT(I, J)	transpose of the load matrix, dimension $I = J \geq$ maximum number of points.
N1 and N2	when adding routes to be improved, the points of the empty link.
NAD	add route indicator, if = 1 previous routes read.
NCL	one link saving indicator, if = 1, one link savings used.
NC2	two link savings indicator, if = 1 two link savings used.
NC3	three link savings indicator, if = 1 three link savings used.
NOART	number of additional routes to be read.
NOPT	number of points in problem.
NPA	indicator to punch inprocess routes, routes punched if = 1.
NPRB	number of problems in the problem set.
NRT	number of routes created.
NSDT	indicator to read distances, if = 1 distance matrix not read.
NSLD	load matrix read indicator, if = 1 load matrix not read, if > 1 load matrix stored.
ORD	computer input device parameter for READ statements.
OTP	parameter for auxillary storage device.
OWT	computer input device parameter used for write statements.
SI(I, J, K)	the savings points corresponding to SV(I, J, K), each field contains four two digit integers, dimension $I = J \geq$ maximum number of points and $K = 3$.

SV(I,J,K) savings for points I and J, if K = 1 contains one link savings, array K = 2 is two link savings, and J = 3 three link savings, dimension I = J \geq maximum number of points and K = 3.

ZRT(I) unloaded links on the route of ART(I,J), each field is subdivided into three four digit integers, dimension I \geq the maximum number of expected routes.

B. COMPUTER PROGRAM

```

C
C      MAIN PROGRAM
      IMPLICIT INTEGER*2(A-N), INTEGER*4(C-T, W-Z)
      COMMON LDT(25,25),DIST(25,25),ART(2000,20),INDEX(25,25,30),
1ZRT(2000),SV(25,25,3),SI(25,25,3),HS(25,25,3),LDX(25,25),AVT(25)
      COMMON NOPT,NRT,NC1,NC2,NC3,NAC,ARTLG
      COMMON AVEL,AWP,DVRT,NFA
      COMMON ORD,CWT
      DIMENSION EL(6),FMT(20)
C      INITIALIZE READ AND WRITE PARAMETERS
      CRD=1
      CWT=3
C      READ FORMAT
      READ(ORD,500) (FMT(I),I=1,20)
500  FORMAT(20A2)
C      READ NO. OF PROBLEMS TO BE EXECUTED AND ECHO IF ECHO = 1
C      INPUT DATA IS PRINTED
      READ(ORD,FMT) NPRB,ECHO
      DO 200 IZ=1,NPRB
C      WRITE PROBLEM NO.
      WRITE(CWT,70) IZ
70  FORMAT('1PROBLEM ',I3)
C      CALL DATA READ ROUTINE
      CALL DATAIN(ECHO,ZLCML)
      CALL TIME(TIM1)
      IF(NC1.EQ.1) GO TO 130
C      PAIR LOADS AND FORM ROUTES
      DO 120 I=1,NCPT
      DO 120 J=1,NOPT
      IF(AVT(J).GT.C) GO TO 116
      CML=0
      GO TO 115
116  CML=LDX(I,J)
      IF(CML.GT.LDX(J,I)) CML=LDX(J,I)
      IF(CML.LE.C) GO TO 115
      IF(CML.GT.AVT(J)) CML=AVT(J)
      AVT(J)=AVT(J)-CML
      NRT=NRT+1
      ZRT(NRT)=0
      ART(NRT,1)=CML
      ART(NRT,2)=2*DIST(I,J)
      ART(NRT,3)=2
      ART(NRT,4)=I
      ART(NRT,5)=J
      ART(NRT,20)=0
C      CREAT MODIFIED TRANSPOSE OF LOAD MATRIX AND UPDATE
C      THE LOAD MATRIX
115  LDT(I,J)=LDX(J,I)-CML
      LDT(J,I)=LDX(I,J)-CML
      LDX(J,I)=LDX(J,I)-CML
      LDX(I,J)=LDX(I,J)-CML
120  CONTINUE
      NS=NRT
      GO TO 145
C      CREAT TRANSPOSE OF LOAD MATRIX
130  DO 140 I=1,NCPT
      DO 140 J=1,NOPT
140  LDT(J,I)=LDX(I,J)

```

```

      NS=1
C      ESTABLISH INITIAL ROUTES
145  DO 155 I=1,NOPT
      DO 155 J=1,NOPT
        IF(AVT(J).LE.0) GO TO 155
        IF(LDT(I,J).LE.0) GO TO 155
        CML=LDT(I,J)
        IF(CML.GT.AVT(J)) CML=AVT(J)
        AVT(J)=AVT(J)-CML
        NRT=NRT+1
        ZRT(NRT)=I*100+J
        ART(NRT,1)=CML
        ART(NRT,2)=2*DIST(I,J)
        ART(NRT,3)=2
        ART(NRT,4)=I
        ART(NRT,5)=J
        ART(NRT,20)=0
        CALL IDXCH(I,J,NRT)
155  CONTINUE
      NNS=NRT
C      CALCULATE INITIAL SAVINGS
      DO 150 I=1,NOPT
        DO 150 J=1,NOPT
          IF(I.EQ.J) GO TO 150
          IF(NC1.EQ.1) CALL SA1(I,J)
          IF(NC2.EQ.1) CALL SAV(I,J)
          IF(NC3.EQ.1) CALL SAVV(I,J)
150  CONTINUE
C      EXPANSION OF ROUTES
      NP=2
      NC=2
      IF(NC1.EQ.1) NP=1
      IF(NC3.EQ.1) NC=3
1000  MMX=0
C      FIND MAXIMUM SAVINGS
      DO 160 K=NP,NC
        DO 160 I=1,NOPT
          DO 160 J=1,NOPT
            MHSV=SV(I,J,K)/1000
            IF(MHSV.LE.MMX) GO TO 160
            MMX=MHSV
            MR1=I
            MR2=J
            MR3=K
            MMR=I*100+J
160  CONTINUE
C      IF MAXIMUM SAVINGS EQUAL ZERO GO TO ROUTE OUTPUT
      IF(MMX.LE.0) GO TO 170
C      CHECK MAXIMUM SAVINGS
      IF(MR3.EQ.1) CALL SA1(MR1,MR2)
      IF(MR3.EQ.2) CALL SAV(MR1,MR2)
      IF(MR3.EQ.3) CALL SAVV(MR1,MR2)
      MHSV=SV(MR1,MR2,MR3)/1000
      IF(MMX.NE.MHSV) GO TO 1000
C      FIND ROUTE WITH CORRECT SAVINGS INDEX
      NTZ=0
      NXS=0
      DO 168 NK=1,30

```



```

      BN=INDEX(MR1,MR2,NK)
      IF(BN.EQ.0) GO TO 165
      NAR=ART(BN,2)
      IF(NAR.GT.2.AND.MR3.EQ.1) GO TO 168
C      CHECK RESTRICTIONS
      CALL RESTR(MR1,MR2,MR3,BN,NSZ)
      IF(MXS-NSZ) 166,168,168
166  NKA=NK
      MXS=NSZ
      NTZ=0
      GO TO 168
165  NTZ=NTZ+1
      IF(NTZ.GT.3) GO TO 167
168  CONTINUE
C      IF LOADS EQUAL ZERO SET SAVINGS EQUAL ZERO
167  IF(MXS.GT.0) GO TO 169
      SV(MR1,MR2,MR3)=0
      GO TO 1000
C      CALL ROUTING SUBROUTINE
169  CALL RTE(NKA,MR1,MR2,MR3)
      GO TO 1000
C      WRITE OUT LF ROUTES
170  CALL TIME(TIM2)
      WRITE(CWT,70) 1Z
      WRITE(CWT,40)
40  FORMAT(' SOLUTION ROUTES')
      WRITE(CWT,50)
50  FORMAT(' ORDER OF ROUTE')
      NO=0
      SML=0
      SEM=0
      AD=0
      DO 190 K=1,NRT
C      CHECK IF ROUTE IS ACTIVE
      IF(ART(K,1).LE.0) GO TO 190
C      DETERMINE NO. OF ARCHS
      LL=ART(K,3)+3
      LP=LL-1
C      CHECK FOR EMPTY ARCHS
      IF(ZRT(K).LE.0) GO TO 180
      ZZ=0
C      ESTABLISH EMPTY ARCH INDICATORS
      DO 175 L=1,6
      LL=L*2
      EL(L)=ZRT(K)/(10**((12-LL)))-ZZ
175  ZZ=(ZZ+EL(L))*100
C      CALCULATE EMPTY LENGTH
      EML=0
      DO 181 L=1,5,2
      IF(EL(L).GT.0) EML=EML+DIST(EL(L),EL(L+1))
181  CONTINUE
      GO TO 185
180  EML=0
      DO 182 L=1,6
182  EL(L)=0
C      WRITE POINTS ON ROUTE
185  WRITE(CWT,55) ART(K,LL),(ART(K,L),L=4,LP)
      55  FORMAT(' 20(2X,13))

```

```

C      CALCULATE AVG. DIST. TRAVELED PER TIME PERIOD
      AVD=AVEL*ALP*(ART(K,20)+1)
      LD=AVD+DVRT*AVEL
      LB=AVD-DVRT*AVEL
C      FIND ROUTE POSITION
      EJ=0
      EI=0
      CD=DIS(ART(K,4),ART(K,LL))
      IF(DC,LI,LC) GO TO 240
      K1=LL
      K2=4
      CC TO 260
240 CC 250 K1=4,LP
      K2=K1+1
      EC=DC+DIS(ART(K,K1),ART(K,K2))
      IF(EL(5).EQ.ART(K,K1)) E1=1
      IF(DG,GE,LD) GO TO 260
250 CONTINUE
260 IF(DC,LE,UC) EJ=1
      IF(EL(5).EQ.0) EI=1
C      WRITE LDS., LGTH., NC. ARCHS, EMPTY LGTH., AND
C      EMPTY ARCHS
      /A=ART(K,20)+1
      WRITE(OWT,56) (ART(K,L),L=1,3),EML,(EL(L),L=3,6)
56 FORMAT(' ', 'LDS='12,3X,'LGTH='16,3X,'NC ARCHS='12,3X,'EMPTY
      LGTH='16,3X,'EMPTY ARCHS',413)
      IF(EJ,EQ,1) GO TO 350
      WRITE(OWT,57) ART(K,K1),ART(K,K2),AA
57 FORMAT(' ',25X,'CARRIER BETWEEN PTS ',213,3X,'TIME PERIOD ',12)
      CC TO 365
350 WRITE(OWT,58) ART(K,K2),AA
58 FORMAT(' ', 'CARRIER AT PT ',13,3X,'TIME PERIOD ',12)
C      KEEP TOTALS OF NC. DIFF. RTS., LGTH., AND EMPTY LGTH.
365 AO=NC+1
      SML=SML+ART(K,1)*ART(K,2)
      SEM=SEM+ART(K,1)*EML
C      IF NPA=1 RTS. TO BE INCLUDED NEXT RUN ARE PUNCHED
      IF(NPA.NE.1.OR.EI.EQ.1) GO TO 190
      AD=AC+1
      WRITE(2,280) EL(5),EL(6),AA
280 FORMAT(8110)
      WRITE(2,280) (ART(K,L),L=1,3)
      WRITE(2,280) (ART(K,L),L=4,LL)
190 CONTINUE
C      PUNCH NO. OF ADDITIONAL ROUTES
      IF(NPA.EQ.1) WRITE(2,280) AC
C      WRITE TOTALS
      WRITE(OWT,20) NC,SML,SEM
20 FORMAT('NUMBER OF TOURS ',14,5X,'TOTAL MILEAGE ',110,5X,'EMPTY MI
      LEAGE ',110)
C      WRITE LOAD MILES
      LTIME=(TIM2-TIM1)/100.
      WRITE(OWT,61) ZLML,UTIME,AC
61 FORMAT('LOAD MILES CAL. ',110,5X,'EXEC. TIME ',F10.2,' SEC.',5X,
      1*NO. IMP. RTS. ',13)
C      UPDATE TRANSPOSE OF LCS MATRIX
      EG 204 I=NS,ANS
      IF(ART(I,3).NE.2) GO TO 204

```

```

      LDT(ART(I,4),ART(I,5))=LDT(ART(I,4),ART(I,5))-ART(I,1)
204 CONTINUE
C      WRITE UNROUTED LCADS
      WRITE(CWT,201)
201 FORMAT('--LIST OF UNROUTED LOADS')
      DO 210 I=1,NCPT
      DO 210 J=1,NCPT
      IF(LDT(I,J).LE.0) GO TO 210
      WRITE(CWT,202) J,I,LLT(I,J)
202 FORMAT(' FROM ',I2,' TO ',I2,5X,I4,' LCADS')
210 CONTINUE
200 CONTINUE
      STOP
      END

```

```

C
SUBROUTINE DATAIN(ECHO,ZLDML)
IMPLICIT INTEGER*2(A-N), INTEGER*4(O-I, W-Z)
COMMON LBT(25,25),DIST(25,25),ART(2000,20),INDEX(25,25,30),
JRT(2000),SV(25,25,3),SI(25,25,3),BS(25,25,3),LDX(25,25),AVT(25)
COMMON NOPT,NRT,NC1,NC2,NC3,NAC,ARTLG
COMMON AVEL,AWP,DVRT,NFA
COMMON GRD,OWT
DIMENSION FMT(20)
C      READ DATA
C      READ FORMAT
      READ(ORD,500)(FMT(K),K=1,20)
500 FORMAT(20A2)
C      READ NO. OF POINTS AND OPTION PARAMETERS
C      IF NSLD =1 LOAD MATRIX NOT READ
C      IF NSLD>1 LOAD MATRIX STORED
C      IF NSDT =1 DISTANCE MATRIX NOT READ
C      IF NPA=1 ROUTES HAVING POSSIBLE OF IMPROVEMENT PUNCHED
      READ(ORD,FMT) NCPT,NC1,NC2,NC3,NAC,NSLD,NSDT,NPA
C      READ MAX. RT, LGTH., AVG. VEL., AVG. TIME PERIOD, VAR. IN TIME ALLOWED
      READ(ORD,FMT) ARTLG,AVEL,AWP,DVRT
C      INITIALIZE TLD, SV, SI, AND INDEX
      DO 95 I=1,NCPT
      DO 95 J=1,NOPT
        LBT(I,J)=0
        DO 90 K=1,3
          SV(I,J,K)=0
          SI(I,J,K)=0
        90 INDEX(I,J,K)=0
        DO 95 K=4,30
          INDEX(I,J,K)=0
        95 INDEX(I,J,K)=0
C      INITIALIZE TAPE DRIVE PARAMETER AND WIND TAPE
      CIP=11
      REWIND OIP
      IF(NSLD.EQ.1) GO TO 102
C      READ FORMAT
      READ(ORD,500)(FMT(K),K=1,20)
C      READ LOADS AT EACH POINT
C      LOAD MATRIX STORED
      DO 100 I=1,NCPT
100 READ(ORD,FMT)(LDX(I,J),J=1,NOPT)
C      STORE LOAD MATRIX
      IF(NSLD.LE.1) GO TO 105
      WRITE(OIP)((LDX(I,K),K=1,NOPT),I=1,NCPT)
      GO TO 105
C      RE-INITIALIZE LOAD MATRIX
102 READ(OIP)((LDX(I,K),K=1,NOPT),I=1,NCPT)
105 IF(NSDT.EQ.1) GO TO 108
C      READ FORMAT
      READ(ORD,500)(FMT(K),K=1,20)
C      READ DISTANCE MATRIX
      NN=NCPT-1
      DO 110 I=1,NC
      NN=I+1
110 READ(ORD,FMT)(DIST(I,J),J=NN,NOPT)
C      COMPLETE DISTANCE MATRIX
      DO 111 I=1,NOPT
      DO 111 J=1,I

```

```

      IF(I.EQ.J) DIST(I,J)=0
111  DIST(I,J)=DIST(J,I)
C      READ FORMAT CARD
108  READ(ORD,500) (FMT(K),K=1,20)
C      READ NO. AND LOCATION OF ICLE CARRIERS
      READ(ORD,FMT) (AVI(K),K=1,NOPT)
C      CALCULATE LOAD MILES
      ZLDML=0
      DO 112 I=1,NOPT
      DO 112 J=1,NOPT
112  ZLDML=LDX(I,J)*DIST(I,J)+ZLDML
      NRT=0
      IF(NAO.NE.1) GO TO 140
C      READ PREVIOUS ROUTES
C      READ FORMAT
      READ(ORD,500) (FMT(K),K=1,20)
C      READ NO. ADDITIONAL ROUTES
      READ(ORD,FMT) NCART
      DO 130 I=1,NCART
      NRT=NRT+1
      ZRT(NRT)=0
C      READ EMPTY ARCH INDICATORS, TIME PERIOD INDICATOR
      READ(ORD,FMT) N1,N2,ART(NRT,20)
C      READ NO. LDS., LGTH., AND NO. ARCHS OF ROUTE
      READ(ORD,FMT) (ART(NRT,J),J=1,3)
      N=ART(NRT,3)+3
C      READ POINTS ON THE ROUTE
      READ(ORD,FMT) (APT(NRT,J),J=4,N)
      ZRT(NRT)=N1*100+N2
      CALL IDXCH(N1,N2,NRT)
130  CONTINUE
140  IF(EC=0.NE.1) RETURN
C      ECFG CHECK OF DATA
      WRITE(CWT,30) NCPT,NC1,NC2,NC3
30  FORMAT('—NUMBER OF POINTS ',I3,5X,'OPTION PARAMETERS ',3(I3))
      WRITE(CWT,39) ARTLG,AVEL,AWP,CVRT
39  FORMAT('—MAX RTE LGTH ',I6,5X,'AVG VEL ',I6,5X,'AVG RUN PERIOD ',I
16,5X,'VAR IN RUN PERIOD ',I6)
      WRITE(CWT,37)
37  FORMAT('—AVAILABLE ICLE CARRIERS AND LOCATION')
      WRITE(CWT,38) (AVI(I),I=1,NCPT)
38  FORMAT(' ',40I3)
      WRITE(CWT,31)
31  FORMAT('—LOAD MATRIX')
      DO 33 I=1,NOPT
33  WRITE(CWT,32) I,(LDX(I,J),J=1,NOPT)
32  FORMAT(' ROW ',I3,5X,30I3)
      WRITE(CWT,34)
34  FORMAT('—DISTANCE MATRIX')
      DO 35 I=1,NCPT
35  WRITE(CWT,36) I,(DIST(I,J),J=1,NCPT)
36  FORMAT(' ROW ',I2,3X,20(IX,I4))
      RETURN
      END

```

```

C      SUBROUTINE RTE(KN,M,K,MR)
C      ROUTING SUBROUTINE
      IMPLICIT INTEGER*2(A-N),INTEGER*4(C-I,W-Z)
      COMMON LDT(25,25),LIST(25,25),ART(2000,20),INDEX(25,25,30),
12RT(2000),SV(25,25,3),SI(25,25,3),ES(25,25,3),LEX(25,25),AVT(25)
      COMMON NOPT,ART,NC1,NC2,NC3,NAB,ARTLG
      COMMON AVEL,AKP,DVRT,KFA
      COMMON ORU,LWT
      DIMENSION AXT(20),CT(4),CT(3),EL(6)
C      ROUTE BEING UP DATED
      K=INDEX(K,N,KN)
C      INITIALIZE UPDATE POINTS
      LL=ART(K,3)+3
      ZI=SI(M,N,MR)
C      SAVINGS POINTS
      CT(1)=ZI/1000000
      CT(2)=ZI/10000-CT(1)*100
      CT(3)=ZI/100-CT(1)*10000-CT(2)*100
      CT(4)=ZI-CT(1)*1000000-CT(2)*10000-CT(3)*100
C      EMPTY ARCHS OF SAVINGS
      CT(1)=BS(M,N,MR)/100
      CT(2)=BS(M,N,MR)/10-CT(1)*10
      CT(3)=BS(M,N,MR)-CT(1)*100-CT(2)*10
C      FIND MINIMUM OF SAVINGS AND ROUTE LOADS
      LCH=SV(M,N,MR)-(SV(M,N,MR)/1000)*1000
      IF(LCH.GT.ART(K,1)) LCH=ART(K,1)
      NNS=LCH
      IF(LCH.GT.ART(K,1).AND.ART(K,3).GT.2) LCH=ART(K,1)
      MSS=ART(K,1)
      CH=CT(1)
      IF(CT(1).EQ.C) CH=CT(2)
      IF(CT(2).EQ.C) CH=CT(3)
C      FIND EMPTY ARCHS FOR ROUTE
      ZZ=C
      DO 190 L=1,6
      LL=L*2
      EL(L)=ZRT(K)/(10**((12-LL))-ZZ
      ZZ=EL(L)*100
      IF(EL(L).EQ.CH) EL(L)=C
190 CONTINUE
C      FIND ROUTE UPDATE POINT
      LL=ART(K,3)+3
      DO 180 I1=4,LL
      NCH=ART(K,I1)
      IF(CH-NCH) 180,100,180
180 CONTINUE
      RETURN
C      STORE POINTS TO RIGHT OF UPDATE POINT
100 IN=I1+2
      IF(IN-LL) 151,151,109
151 DO 110 I2=IN,LL
110 AXT(I2)=ART(K,I2)
C      UPDATE ROUTE NUMBER
109 IF(NNS-MSS) 111,107,107
107 KK=K
      INDEX(M,N,KN)=C
      GO TO 112

```

```

111 NRT=NRT+1
    KK=NRT
C      UPDATE ROUTE LOADS
    ART(K,1)=PSS-LCH
112 ART(KK,1)=LCH
    ART(KK,20)=ART(K,20)
    IF(1-MP) 161,145,145
C      TRANSFER ROUTE POINTS
161 GO 113 LX1=4,I1
113 ART(KK,LX1)=ART(K,LX1)
    IF(CI(1)) 160,162,160
162 NS=1
    ART(KK,2)=ART(K,2)+DIST(CI(2),CI(3))+DIST(CI(3),CI(4))-DIST(CI(2),
    CI(4))
C      UPDATE ROUTE POINTS
    ART(KK,3)=ART(K,3)+1
    GO 117 I=2,4
    II=I1+I-2
117 ART(KK,II)=CI(I)
    GO TO 170
160 NS=2
    ART(KK,2)=ART(K,2)+DIST(CI(1),CI(2))+DIST(CI(2),CI(3))+DIST(CI(3),
    CI(4))-DIST(CI(1),CI(4))
    ART(KK,3)=ART(K,3)+2
    GO 116 I=1,4
    II=I1+I-1
116 ART(KK,II)=CI(I)
C      ADD RIGHT POINTS
170 IF(IK-LL) 163,163,108
163 GO 114 I3=IN,LL
114 ART(KK,I3+NS)=AXT(I3)
C      UPDATE NET TRANSPOSE MATRIX AND SAVINGS MATRICES
108 ZRT(KK)=0
    LDT(CI(3),CI(2))=LDT(CI(3),CI(2))-LCH
    LDT(CI(4),CI(3))=LDT(CI(4),CI(3))-LCH
    IF(CI(1),EQ,C) LDT(CI(2),CI(4))=LDT(CI(2),CI(4))-LCH
    IF(CI(1)) 164,141,164
164 LDT(CI(2),CI(1))=LDT(CI(2),CI(1))-LCH
    LDT(CI(1),CI(4))=LDT(CI(1),CI(4))-LCH
C      UPDATE EMPTY ARCH INDICATORS
141 GO 120 I=1,3
    IF(CI(I)-1) 120,115,120
115 GO 197 L=1,5,2
    IF(EL(L).NE.C) GO TO 197
    EL(L)=CI(I)
    EL(L+1)=CI(I+1)
    CALL IDXC(CI(I),CI(I+1),KK)
    IF(NC2.EQ.1) CALL SAV(CI(I),CI(I+1))
    IF(NC3.EQ.1) CALL SAVV(CI(I),CI(I+1))
    GO TO 120
197 CONTINUE
120 CONTINUE
C      STORE EMPTY ARCH INDICATORS
    ZRT(KK)=0
    GO 198 I=1,5,2
    IF(EL(I).EQ.C) GO TO 198
    ZRT(KK)=ZRT(KK)+10000+EL(I)*100+EL(I+1)
198 CONTINUE

```

```

      GO TO 148
145 ZRT(KK)=0
      DO 147 I2=2,5
147 ART(KK,I2)=ART(K,I2)
      LCT(CT(3),CT(4))=LDT(CT(3),CT(4))-LCH
      LDT(CT(4),CT(3))=LDT(CT(4),CT(3))-LCH
148 IF(NC1.NE.1)GO TO 149
      CALL SA1(N,M)
      CALL SA1(M,N)
149 IF(NC3.EQ.1) CALL SAVV(M,N)
      IF(NC2.EQ.1 ) CALL SAV(M,N)
C      UPDATE OLD ROUTES
      DO 150 KP=1,3
      IF(CT(KP).EQ.0.CR.CT(KP+1).EQ.0.CR.CT(KP).EQ.1) GO TO 150
      PP=0
200 PP=PP+1
      NCK=INDEX(CT(KP+1),CT(KP),PP)
      IF(NCK.LE.0) GO TO 150
      IF(ART(NCK,20).GT.0) GO TO 200
      IF(ART(NCK,3)-2) 150,165,150
165 ART(NCK,1)=ART(NCK,1)-LCH
      IF(ART(NCK,1)) 150,166,150
166 ZRT(NCK)=0
150 CONTINUE
      RETURN
      END

```



```

C      SUBROUTINE SAI(I,J)
        IMPLICIT INTEGER*2(A-N),INTEGER*4(C-T,W-Z)
        COMMON LDT(25,25),DIST(25,25),ART(2000,20),INDEX(25,25,30),
1       IZRT(2000),SV(25,25,3),SI(25,25,3),BS(25,25,3),LDX(25,25),AVI(25)
        COMMON NOPT,ARI,AC1,AC2,AC3,NAC,ARILG
        COMMON AVEL,AKP,EVRT,NPA
        COMMON DRD,CWF
C      CALCULATE SAVINGS FROM PAIRING
        CPL=LDI(I,J)
C      UPDATE SAVINGS MATRICES
        SV(I,J,1)=DIST(I,J)*1000+CPL
        SI(I,J,1)=I*100+J
        BS(I,J,1)=0
        IF(CPL.LE.0) SV(I,J,1)=0
        RETURN
        END

```

```

C      SUBROUTINE SAV(I,J)
C      SAVINGS SUBROUTINE
      IMPLICIT INTEGER*2(A-N), INTEGER*4(C-T, W-Z)
      COMMON LDT(25,25),DIST(25,25),ART(2000,20),INDEX(25,25,30),
1ZRT(2000),SV(25,25,3),SI(25,25,3),BS(25,25,3),LDX(25,25),AVT(25)
      COMMON NOPT,NRT,NC1,NC2,NC3,NAD,ARTLG
      COMMON AVEL,AKP,CVRT,NPA
      COMMON ORD,CWT
C      CALCULATION OF SAVINGS TRIANGLE METHOD
      MMX=0
      MD=0
      NN1=0
      NN2=0
      NKK=0
      IF(LDT(I,J).EQ.0.AND.INDEX(I,J,1).LE.0) GO TO 130
100  DO 120 K=1,NOPT
      IF(K.EQ.I.OR.K.EQ.J) GO TO 120
      ND1=0
      ND2=0
      AD=9999
      AS=DIST(I,J)
      CALL SSAV(AS,AD,K,I,ND1)
      CALL SSAV(AS,AD,J,K,ND2)
      IF(AD.LE.0.OR.AS.LE.MMX) GO TO 120
110  MMX=AS
      MD=AD
      NN1=ND1
      NN2=ND2
      NKK=K
120  CONTINUE
C      UPDATE SAVINGS MATRICES
130  SI(I,J,2)=I*10000+NKK*100+J
      BS(I,J,2)=NN1*10+NN2
      SV(I,J,2)=MMX*1000+MD
      RETURN
      END

```

```

C
  SUBROUTINE SAVV(I,J)
    IMPLICIT INTEGER*2(A-N),INTEGER*4(C-I,K-Z)
    COMMON LDT(25,25),DIST(25,25),ART(2000,2),INDEX(25,25,30),
1/RT(2000),SV(25,25,3),SI(25,25,3),BS(25,25,3),LCX(25,25),AVT(25)
    COMMON NUPT,ART,NC1,NC2,NC3,KAC,ARILG
    COMMON AVEL,AKP,CVRT,NFA
    COMMON ORD,CRT
C
    CALCULATE SAVINGS USING TRAPEZOID METHOD
    NK=0
    NL=0
    ND=0
    MPX=0
    NN1=0
    NN2=0
    NN3=0
    IF(LDT(I,J).EQ.0.AND.INDEX(I,J,1).LE.0) GO TO 120
200 GO 110 K=1,NUPT
    IF(K.EQ.1.OR.K.EQ.J) GO TO 110
    DO 100 L=1,NCPT
    IF(L.EQ.1.OR.L.EQ.J.OR.L.EQ.K) GO TO 100
    ND1=0
    ND2=0
    ND3=0
    AS=DIST(I,J)
    AD=9999
    CALL SSAV(AS,AD,K,I,ND1)
    CALL SSAV(AS,AD,L,K,ND2)
    CALL SSAV(AS,AD,J,L,ND3)
    IF(AD.LE.0.CR.AS.LE.MPX) GO TO 100
190 MPX=AS
    ND=AD
    NK=K
    NL=L
    NN1=ND1
    NN2=ND2
    NN3=ND3
100 CONTINUE
110 CONTINUE
C
    UPDATE SAVINGS MATRICES
120 SI(I,J,3)=I*1000000+NK*10000+NL*100+J
    PS(I,J,3)=NN1*100+NN2*10+NN3
    SV(I,J,3)=MPX*1000+ND
    RETURN
  END

```

```

C
  SUBROUTINE SSAV(AS,AD,N,P,NA)
    IMPLICIT INTEGER*2(A-N),INTEGER*4(C-T,W-Z)
    COMMON LDT(25,25),DIST(25,25),ART(2030,20),INDEX(25,25,30),
12RT(2000),SV(25,25,3),SI(25,25,3),BS(25,25,3),LEX(25,25),AVT(25)
    COMMON NORT,NRT,NC1,NC2,NC3,NAD,ARTLG
    COMMON AVEL,ALP,EVRT,IFA
    COMMON ORD,CWT
C      TABULATE SAVINGS AVAILABLE IF PTS. N AND P INCLUDED
    IF(LDT(N,P).GT.0) GO TO 100
    AS=AS-DIST(N,P)
    NR=1
    RETURN
100 AS=AS+DIST(N,P)
    IF(AD.GT.LDT(N,P)) AD=LDT(N,P)
    RETURN
  END

```

C

```

      SUBROUTINE IUXCH(I,J,NCRT)
      IMPLICIT INTEGER*2(A-N),INTEGER*4(O-T,W-Z)
      COMMON LDI(25,25),DIST(25,25),ART(2000,20),INDEX(25,25,30),
      1ZRT(2000),SV(25,25,3),SI(25,25,3),RS(25,25,3),LDX(25,25),AVI(25)
      COMMON NDPT,NRT,NC1,NC2,NC3,NAC,ARTLG
      COMMON AVEL,AWP,CVRT,NFA
      COMMON ORD,CWT

```

C

```

      ESTABLISH INDEX FOR SAVINGS POINTS ON ROUTE
      NC=30
      EG 100 K=1,NC
      IF(INDEX(I,J,K)) 100,110,100
100 CONTINUE
      WRITE(CWT,10)
      10 FORMAT('OINDEX EXCEEDED RANGE')
      RETURN
110 INDEX(I,J,K)=NCRT
      RETURN
      END

```

```

C      SUBROUTINE RESTR(I,J,K,IX,NSZ)
        IMPLICIT INTEGER*2(A-N),INTEGER*4(C-T,W-Z)
        COMMON LDT(25,25),DIST(25,25),ART(2000,20),INDEX(25,25,30),
1       IZRT(2000),SV(25,25,3),SI(25,25,3),BS(25,25,3),LDX(25,25),AVT(25)
        COMMON NOPT,NRT,NC1,NC2,NC3,NAC,ARTLG
        COMMON AVEL,AWP,DVRT,NFA
        COMMON ORD,CWT
C      RESTRICTIONS ON ROUTES
        NSZ=ART(IX,1)
        ZI=SI(I,J,K)-(SI(I,J,K)/1000000)*1000000
        C1=ZI/10000
        C2=ZI/100-C1*100
        C3=ZI-C1*10000-C2*100
        IF(C1.EQ.0) GO TO 10
        ED=ART(IX,2)+DIST(C1,C2)+DIST(C2,C3)-DIST(C1,C3)
        GO TO 20
10     ED=ART(IX,2)
20     IF(UL.GT.ARTLG) NSZ=0
        RETURN
C      END

```

C. SAMPLE COMPUTER OUTPUT

PROBLEM 1

NUMBER OF POINTS 10 OPTION PARAMETERS 1 1 0

MAX RTE LGTH 1600 AVG VEL 60 AVG RUN PERIOD 8

AVAILABLE IDLE CARRIERS AND LOCATION

99 99 99 99 99 99 99 99 99 99

LOAD MATRIX

RCW 1	0	1	0	1	2	0	3	4	1	2
RCW 2	2	0	1	4	0	1	1	1	3	2
RCW 3	1	2	0	0	3	2	1	1	1	2
RCW 4	2	1	1	0	2	2	1	0	0	1
RCW 5	3	0	4	1	0	2	0	2	3	1
RCW 6	1	2	0	0	1	0	0	1	1	2
RCW 7	1	1	1	0	1	1	0	3	0	1
RCW 8	2	0	2	2	1	1	2	0	1	0
RCW 9	1	0	0	2	1	0	0	1	0	2
RCW 10	0	1	2	3	0	0	5	0	1	0

DISTANCE MATRIX

ROW 1	0	85	110	165	80	220	155	235	155	215
ROW 2	85	0	195	80	150	135	70	150	130	250
ROW 3	110	195	0	270	80	290	160	240	155	120
ROW 4	165	80	270	0	190	55	110	160	170	290
ROW 5	80	150	80	190	0	210	80	160	75	135
ROW 6	220	135	290	55	210	0	130	105	190	310
ROW 7	155	70	160	110	80	130	0	80	60	180
ROW 8	235	150	240	160	160	105	80	0	105	225
ROW 9	155	130	155	170	75	190	60	105	0	120
ROW 10	215	250	120	290	135	310	180	225	120	0

PROBLEM 1
SOLUTION ROUTES
ORDER OF ROUTE

6	3	1	8	LDS= 1	LGTH= 1349	NO ARCHS= 4	EMPTY LGTH= 0	EMPTY ARCHS	0	0	0	0
				CARRIER AT PT	6	TIME PERIOD 3						
2	7	9	3	LDS= 1	LGTH= 1480	NO ARCHS= 5	EMPTY LGTH= 155	EMPTY ARCHS	0	0	3	9
				CARRIER AT PT	2	TIME PERIOD 3						
7	1	8		LDS= 1	LGTH= 1198	NO ARCHS= 3	EMPTY LGTH= 0	EMPTY ARCHS	0	0	0	0
				CARRIER AT PT	7	TIME PERIOD 2						
6	10	2		LDS= 1	LGTH= 935	NO ARCHS= 3	EMPTY LGTH= 0	EMPTY ARCHS	0	0	0	0
				CARRIER AT PT	6	TIME PERIOD 2						
7	10	9		LDS= 1	LGTH= 1055	NO ARCHS= 3	EMPTY LGTH= 60	EMPTY ARCHS	0	0	9	7
				CARRIER AT PT	7	TIME PERIOD 2						
9	2	3	8	LDS= 1	LGTH= 884	NO ARCHS= 4	EMPTY LGTH= 0	EMPTY ARCHS	0	0	0	0
				CARRIER AT PT	9	TIME PERIOD 2						
2	1	4		LDS= 1	LGTH= 330	NO ARCHS= 3	EMPTY LGTH= 80	EMPTY ARCHS	0	0	4	2
				CARRIER AT PT	2	TIME PERIOD 1						
2	1	8		LDS= 1	LGTH= 585	NO ARCHS= 3	EMPTY LGTH= 0	EMPTY ARCHS	0	0	0	0
				CARRIER BETWEEN PTS	8	3	TIME PERIOD 1					
4	1	10		LDS= 2	LGTH= 670	NO ARCHS= 3	EMPTY LGTH= 0	EMPTY ARCHS	0	0	0	0
				CARRIER BETWEEN PTS	10	4	TIME PERIOD 1					
5	1			LDS= 1	LGTH= 160	NO ARCHS= 2	EMPTY LGTH= 0	EMPTY ARCHS	0	0	0	0
				CARRIER AT PT	5	TIME PERIOD 1						
6	1			LDS= 1	LGTH= 440	NO ARCHS= 2	EMPTY LGTH= 220	EMPTY ARCHS	0	0	1	6
				CARRIER AT PT	6	TIME PERIOD 1						
7	1			LDS= 1	LGTH= 310	NO ARCHS= 2	EMPTY LGTH= 0	EMPTY ARCHS	0	0	0	0
				CARRIER AT PT	7	TIME PERIOD 1						
6	1	7		LDS= 1	LGTH= 470	NO ARCHS= 3	EMPTY LGTH= 0	EMPTY ARCHS	0	0	0	0
				CARRIER AT PT	8	TIME PERIOD 1						
9	1	2		LDS= 1	LGTH= 370	NO ARCHS= 3	EMPTY LGTH= 0	EMPTY ARCHS	0	0	0	0
				CARRIER AT PT	9	TIME PERIOD 1						
4	2	9		LDS= 1	LGTH= 380	NO ARCHS= 3	EMPTY LGTH= 0	EMPTY ARCHS	0	0	0	0
				CARRIER AT PT	4	TIME PERIOD 1						
6	2	4		LDS= 1	LGTH= 270	NO ARCHS= 3	EMPTY LGTH= 0	EMPTY ARCHS	0	0	0	0
				CARRIER AT PT	6	TIME PERIOD 1						
7	2	10		LDS= 1	LGTH= 500	NO ARCHS= 3	EMPTY LGTH= 0	EMPTY ARCHS	0	0	0	0
				CARRIER AT PT	7	TIME PERIOD 1						
2	3	7		LDS= 1	LGTH= 425	NO ARCHS= 3	EMPTY LGTH= 70	EMPTY ARCHS	0	0	7	2
				CARRIER AT PT	2	TIME PERIOD 1						
4	3	10		LDS= 1	LGTH= 680	NO ARCHS= 3	EMPTY LGTH= 0	EMPTY ARCHS	0	0	0	0
				CARRIER BETWEEN PTS	10	4	TIME PERIOD 1					

5	3	LDS= 2	LGTH= 160	NO ARCHS= 2	EMPTY LGTH= 0	EMPTY ARCHS	0	0	0	0
		CARRIER AT PT	5	TIME PERIOD 1						
7	3	10	LDS= 1	LGTH= 460	NO ARCHS= 3	EMPTY LGTH= 0	EMPTY ARCHS	0	0	0
			CARRIER AT PT	7	TIME PERIOD 1					
10	3	6	LDS= 2	LGTH= 720	NO ARCHS= 3	EMPTY LGTH= 0	EMPTY ARCHS	0	0	0
				CARRIER BETWEEN PTS	6 10	TIME PERIOD	1			
2	4	LDS= 2	LGTH= 160	NO ARCHS= 2	EMPTY LGTH= 80	EMPTY ARCHS	0	0	4	2
		CARRIER AT PT	2	TIME PERIOD 1						
5	4	6	LDS= 1	LGTH= 455	NO ARCHS= 3	EMPTY LGTH= 0	EMPTY ARCHS	0	0	0
			CARRIER AT PT	5	TIME PERIOD 1					
8	4	5	LDS= 2	LGTH= 510	NO ARCHS= 3	EMPTY LGTH= 0	EMPTY ARCHS	0	0	0
			CARRIER AT PT	8	TIME PERIOD 1					
7	5	10	LDS= 1	LGTH= 395	NO ARCHS= 3	EMPTY LGTH= 0	EMPTY ARCHS	0	0	0
			CARRIER AT PT	7	TIME PERIOD 1					
5	5	6	LDS= 1	LGTH= 475	NO ARCHS= 3	EMPTY LGTH= 0	EMPTY ARCHS	0	0	0
			CARRIER AT PT	9	TIME PERIOD 1					
5	6	8	LDS= 1	LGTH= 475	NO ARCHS= 4	EMPTY LGTH= 80	EMPTY ARCHS	0	0	7
			CARRIER AT PT	5	TIME PERIOD 1					
7	8		LDS= 2	LGTH= 160	NO ARCHS= 2	EMPTY LGTH= 80	EMPTY ARCHS	0	0	8
			CARRIER AT PT	7	TIME PERIOD 1					
2	9		LDS= 1	LGTH= 260	NO ARCHS= 2	EMPTY LGTH= 130	EMPTY ARCHS	0	0	9
			CARRIER AT PT	2	TIME PERIOD 1					
5	9		LDS= 2	LGTH= 150	NO ARCHS= 2	EMPTY LGTH= 75	EMPTY ARCHS	0	0	9
			CARRIER AT PT	5	TIME PERIOD 1					
3	2	8	LDS= 1	LGTH= 585	NO ARCHS= 3	EMPTY LGTH= 0	EMPTY ARCHS	0	0	0
				CARRIER BETWEEN PTS	8 3	TIME PERIOD	1			
5	4	10	LDS= 1	LGTH= 580	NO ARCHS= 3	EMPTY LGTH= 0	EMPTY ARCHS	0	0	0
			CARRIER AT PT	10	TIME PERIOD 1					
8	1	9	LDS= 1	LGTH= 495	NO ARCHS= 3	EMPTY LGTH= 0	EMPTY ARCHS	0	0	0
			CARRIER AT PT	8	TIME PERIOD 1					
5	1	8	LDS= 1	LGTH= 475	NO ARCHS= 3	EMPTY LGTH= 0	EMPTY ARCHS	0	0	0
			CARRIER AT PT	5	TIME PERIOD 1					
10	7	2	LDS= 1	LGTH= 500	NO ARCHS= 3	EMPTY LGTH= 70	EMPTY ARCHS	0	0	7
			CARRIER AT PT	10	TIME PERIOD 1					
7	10	9	LDS= 1	LGTH= 1295	NO ARCHS= 4	EMPTY LGTH= 0	EMPTY ARCHS	0	0	0
			CARRIER AT PT	7	TIME PERIOD 2					
6	2	7	LDS= 1	LGTH= 335	NO ARCHS= 3	EMPTY LGTH= 0	EMPTY ARCHS	0	0	0
			CARRIER AT PT	6	TIME PERIOD 1					
2	1	5	LDS= 1	LGTH= 440	NO ARCHS= 4	EMPTY LGTH= 0	EMPTY ARCHS	0	0	0
			CARRIER AT PT	2	TIME PERIOD 1					

5	3	9							
LDS= 1	LGTH=	310	NO ARCHS= 3	EMPTY LGTH=	75	EMPTY ARCHS	0	0	9 5
CARRIER AT PT	5	TIME PERIOD	1						
5	1	7	10 3						
LDS= 1	LGTH=	615	NO ARCHS= 5	EMPTY LGTH=	120	EMPTY ARCHS	0	0	10 3
CARRIER AT PT	3	TIME PERIOD	1						
2	4	7							
LDS= 1	LGTH=	260	NO ARCHS= 3	EMPTY LGTH=	70	EMPTY ARCHS	0	0	7 2
CARRIER AT PT	2	TIME PERIOD	1						
5	9	10							
LDS= 1	LGTH=	330	NO ARCHS= 3	EMPTY LGTH=	135	EMPTY ARCHS	0	0	10 5
CARRIER AT PT	5	TIME PERIOD	1						

NUMBER OF TOURS	43	TOTAL MILEAGE	25621	EMPTY MILEAGE	1735
LOAD PILES CAL.	17590	EXEC. TIME	13.66 SEC.	NO. IMP. RTS.	0

LIST OF UNROUTED LOADS

D. SAMPLE DATA SET

(1615)

1 1

(1615)

10 1 1 0 1 0 0 0

1600 60 8 1

(4012)

0 1 0 1 2 0 3 4 1 2

2 0 1 4 0 1 1 1 3 2

1 2 0 0 3 2 1 1 1 2

2 1 1 0 2 2 1 0 0 1

3 0 4 1 0 2 0 2 3 1

1 2 0 0 1 0 0 1 1 2

1 1 1 0 1 1 0 3 0 1

2 0 2 2 1 1 2 0 1 0

1 0 0 2 1 0 0 1 0 2

0 1 2 3 0 0 5 0 1 0

(1615)

85 110 165 80 220 155 235 155 215

195 80 150 135 70 150 130 250

270 80 290 160 240 155 120

190 55 110 160 170 290

210 80 160 75 135

130 105 190 310

80 60 180

105 225

120

(1615)

99 99 99 99 99 99 99 99 99 99

(8110)

6

1

6

2

1

1229

3

3

1

6

3

9

2

1

1480

5

7

9

3

9

2

1

7

1

1

1038

2

1

7

10

6

1

1

860

2

10

6

9

7

1

2

1055

3

10

9

7

3

9

1

1

694

3

2

3

9

A HEURISTIC APPROACH
TO THE LINE HAUL PROBLEM

by

ARMOND DALE SMITH

B.S., Fort Hays Kansas State College, 1969

AN ABSTRACT OF A MASTER'S THESIS

submitted in partial fulfillment of the
requirements for the degree

MASTER OF SCIENCE

Department of Industrial Engineering

KANSAS STATE UNIVERSITY

Manhattan, Kansas

1971

The line haul routing problem was analyzed and possible solution methods evaluated. A heuristic algorithm was developed using the idea of saving to minimize the backhauls required. Three different savings systems were programmed and evaluated. These were initial pairing and two link savings, one and two link savings, and one, two, and three link savings. The one and two link savings proved the most satisfactory.

Modifications were made to the procedure to include a dispatching routine. These changes included a procedure for entering inprocess routes into the present routing period, permitting restrictions on the system, and limiting carrier availability when forming routes. A number of problems were successfully solved with the algorithm.