1970

Heuristic algorithms for the generalized vehicle dispatch problem

Leland Ray Miller

HEURISTIC ALGORITHMS FOR THE GENERALIZED

VEHICLE DISPATCH PROBLEM

by

LELAND RAY MILLER, 1938-

A DISSERTATION

Presented to the Faculty of the Graduate School of the

UNIVERSITY OF MISSOURI-ROLLA

In Partial Fulfillment of the Requirements for the Degree

DOCTOR OF PHILOSOPHY

in

MATHEMATICS

1970

B E Gillett
Advisor

A K Rigler

Ralph E. Lee

Robert W. Eppler

Lee J Bain

ABSTRACT

A heuristic algorithm, called the sweep algorithm, is developed for the vehicle dispatch problem with distance and load constraints for each vehicle. A mathematical development and a step procedure for the sweep algorithm is given. Also given are eight problems and their solutions derived by the sweep algorithm. The solutions for this algorithm are compared with solutions from other vehicle dispatch algorithms, and the sweep algorithm is found to give better results for almost every problem. Various modifications are also presented for the sweep algorithm.

A mathematical formulation is given for the vehicle dispatch problem with arbitrary cost functions at each location. A branch and bound algorithm is developed, which yields an optimal solution for the problem with one server.

ACKNOWLEDGEMENTS

The author wishes to express his sincere appreciation to Dr. Billy E. Gillett for his aid in the selection of this thesis subject and his guidance in the preparation of this dissertation.

The author also wishes to express his thanks to his wife and family for their understanding, encouragement, and sacrifices during these years of graduate study.

TABLE OF CONTENTS

Table of contents (continued) Page

LIST OF ILLUSTRATIONS

LIST OF TABLES

I. INTRODUCTION

There exist many problems that fall into the general category of vehicle dispatch problems; however, there does not exist a simple algorithm which will solve these problems. These problems assume that each of N customers has a given location and demand, and that each location must be serviced by a server. The objective is to determine the minimum number of servers and the routes for each server, so that the total distance that the servers travel is a minimum. Each server is also subject to a load and a distance constraint.

Examples of the problem arise in the delivery of people or commodities such as bread and furniture. These problems assume a known demand. Examples of the problem also arise in scheduling routes such as those for school busses and refuse trucks, where people or commodities are picked up.

It is usually very difficult to determine an exact optimal solution for a problem involving many locations, due to the large number of possible routes that must be examined. Hence, heuristic algorithms have been developed which yield solutions which are hopefully close to an optimal solution. One objective of this paper is to determine a good heuristic algorithm for the vehicle dispatch problem.

A special case of the vehicle dispatch problem is the traveling salesman problem. This case occurs if there are no load and distance restrictions for the servers. Hence, one server is able to meet all the requirements of the customers. The review of literature presents several algorithms for the traveling salesman problem and how they are generalized for the vehicle dispatch problem.

The vehicle dispatch problem can be generalized to include arbitrary cost functions at each location. This creates an additional cost which must be minimized. An example is the scheduling of delivery trucks where a commodity must be delivered in a given time period. This paper presents an exact algorithm which solves the generalized vehicle dispatch problem for one server.

II. REVIEW OF LITERATURE

A. TRAVELING SALESMAN PROBLEM

If there is only one server with no constraints and no arbitrary cost functions, then the vehicle dispatch problem becomes the well-known traveling salesman problem. This problem is that of finding a permutation, $i_2, i_3, \cdots, i_N$ of the integers 2 through N, so that the quantity $a_{1i_2} + \sum_{k=2}^{N-1} (a_{i_k i_{k+1}}) + a_{i_N 1}$ is a minimum. The element a_{ij} could represent either the distance or the time of travel from location i to location j. The name given to the problem is derived from the application of a salesman who wishes to visit N - 1 cities, starting from and returning to his home, by means of the shortest route. This problem was first posed by Hassler Whitney in 1934 [1].

1. Complete Enumeration

There exist a finite number of routes for the salesman, namely (N - 1)!. Therefore, it is theoretically possible to solve the problem by calculating the distance for each of the routes and selecting the route with the minimum distance. However, even for ten locations the number of possible routes is very large, which makes it impossible for a computer to calculate all the distances in any reasonable length of time. For this reason, algorithms have been developed which reduce the calculation time.

2. Dynamic Programming

Dynamic programming was applied to the problem in two articles, each developed independently of the other. One is by Held and Carp [2], and the other is by Bellman [3].

The procedure for dynamic programming is as follows: Let N denote the number of locations and a_{ij} the distance from location i to location j.

For any subset, S, of $\{2, 3, \cdots, N\}$ and $p \varepsilon S$, let $C(S, p)$ represent the minimum distance for starting from location one, visiting all cities in S, and ending at location p. Then a recursive formulation can be given by the following equations:

If $n(S) = 1$, then $C(\{p\}, p) = a_{1p}$ for all $p \varepsilon S$.

If $n(S) > 1$, then $C(S, p) = \min\limits_{m \varepsilon S-p} [C(S-p, m) + a_{mp}]$.

In these equations, $n(S)$ is the cardinality of set S and S-p denotes the set S with the element p omitted. These equations provide a method for calculating $C(S, p)$ inductively, first with $n(S) = 1$, then with $n(S) = 2$, and up to $S = \{2, 3, \cdots, N\}$. The minimum distance of a complete tour, including the return to location 1 is

$$\min_{p \varepsilon \{2, 3, \cdots, N\}} [C(\{2, 3, \cdots, N\}, p) + a_{p1}].$$

The route which yields this minimum distance is obtained by a "backward" procedure. The p_1 which gave the minimum value for $C(\{2, 3, \cdots, N\}, p_1) + a_{p_1 1}$ is the last location on the route. The p_2 which minimizes $C(\{2, 3, \cdots, N\} - \{p_1\}, p_2)$ is the next-to-the-last location on the route. By continuing this procedure until $n(S) = 1$, the route that minimizes the total distance is obtained.

The algorithm requires a large amount of core storage which restricts the size of the problem. Bellmore and Nemhauser [4] were able to solve a 15-location problem using auxiliary storage.

3. Branch and Bound Algorithm

The branch and bound algorithm is also an exact procedure, in that if a solution is obtained it is a route which produces the minimum total distance. The algorithm has been known to solve a 68-location problem; however, it does not propose to solve all problems of this size within a reasonable time limit. Two papers that have been presented which employ the branch and bound method are Shapiro [5] and Little, Murty, Sweeney, and Karel [6].

The basic method of the algorithm is to divide the set of all tours into smaller subsets and to calculate a lower bound for all the tours in the subset. A tree is built with nodes which represent the subsets of tours. Each node is a subset of the node from which it branches. For example, referring to figure 1, node A represents the set of all tours. Node B represents the set of all tours which contain the link i to j. Node C represents the set of all

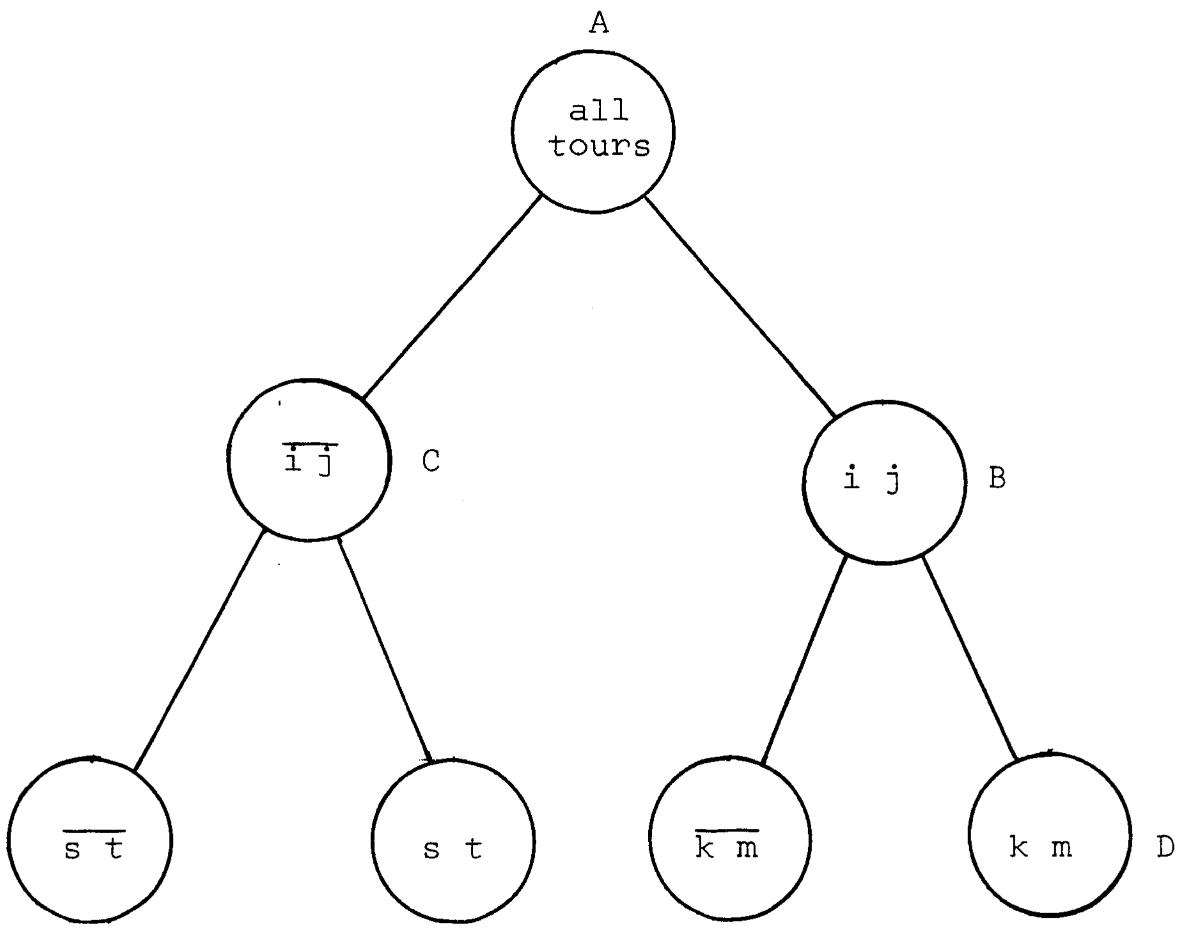

Figure 1. Branches of a tree

tours which do not contain the link i to j. Node D represents the set of all tours containing the links i to j and k to m.

A method similar to the assignment problem is used to calculate lower bounds for each node. The distance matrix, M, is a matrix such that m_{ij} denotes the distance from location i to location j. The lower bound for node A is calculated from the distance matrix using the following theorem: If a constant, h, is subtracted from each element of a row of the distance matrix, then the distance of any tour under the new matrix is h less than under the old. Let r_i be the smallest element in row i $(i = 1, 2, \cdots, N)$ of the distance matrix. The new distance matrix is obtained by subtracting r_i from every element in the i^{th} row for $i = 1, 2, \cdots, N$. The same procedure is used for the columns where c_i is the smallest element in column i $(i = 1, 2, \cdots, N)$. The lower bound is $\sum_{i=1}^{N} (r_i + c_i)$.

Consider the distance matrix for a five-location problem in figure 2. The numbers 2, 2, 1, 2, and 1 are subtracted from rows 1 through 5 respectively, which gives the distance matrix in figure 3. The number 1 is subtracted from columns 1 and 3, which yields the distance matrix in figure 4. The sum of the numbers subtracted from the rows and columns provides the lower bound for node A, which in this example is 10.

	1	2	3	4	5
1	∞	3	3	2	8
2	3	∞	6	4	2
3	8	6	∞	1	4
4	3	2	6	∞	4
5	4	1	8	2	∞

Figure 2. Distance matrix

	1	2	3	4	5	
1	∞	1	1	0	6	(2)
2	1	∞	4	2	0	(2)
3	7	5	∞	0	3	(1)
4	1	0	4	∞	2	(2)
5	3	0	7	1	∞	(1)

Figure 3. Distance matrix with smallest element subtracted from each row

	1	2	3	4	5
1	∞	1	0	0	6
2	0	∞	3	2	0
3	6	5	∞	0	3
4	0	0	3	∞	2
5	2	0	6	1	∞
	(1)		(1)		

Figure 4. Distance matrix with smallest element subtracted from each column

The link i to j for node B is obtained by choosing a zero in the new distance matrix, which will provide the largest lower bound for node C. This is accomplished by placing ∞ in a zero slot, and calculating the lower bound by adding the smallest element in that row, the smallest element in that column, and the previous lower bound. In figure 4, there are 8 zeroes that need to be considered. The link (1,3) provides a lower bound of 10 + 3, since there is a zero in column 4 of row 1 and a three in row 2 of column 3. Likewise, the link (1,4) provides a lower bound of 10 + 0. After all zeroes are checked, (1,3) and (2,5) both are found to provide the greatest bound for C, namely 13.

The link (i,j) which produces the greatest lower bound for node C will be the link used in node B. The lower bound for node B is obtained by omitting row i and column j and calculating $\sum_{\substack{k=1 \\ k\neq i}}^{N} r_k + \sum_{\substack{k=1 \\ k\neq j}}^{N} c_k$, where again, r_k is the smallest element in row k, and c_k is the smallest element in column k.

Branching is continued from the node with the smallest lower bound until all links are used. The lower bound of the last node is the total distance for that particular tour and provides a bound for all other tours. Branching from a node ceases if the lower bound for the node is greater than the smallest bound obtained from the completed tours.

Care must be taken in selecting the link i to j so as to prevent a subtour. Infinity is placed in the slot to prevent this. For example, if (a,b) and (b,c) are two links in previous nodes, then links (a,c) and (c,a) are assigned a distance of infinity.

Computing time varies with each problem, depending on whether a good lower bound which will eliminate many branches is determined at first.

4. Integer Programming

Bellmore and Nemhauser [4] state a theorem which shows that the traveling salesman problem can be set up as a 0 - 1 integer linear-programming problem. The theorem is as follows:

Let S, $\overline{S}$ be a partition of the integers i = 1, 2, $\cdots$, N. An optimal tour can be found by solving the integer linear program;

$$\min z = \sum_{j=2}^{N} \sum_{i=1}^{j-1} a_{ij}x_{ij}$$

subject to;

1. $x_{ij} = 0,1 \; (i = 1,2,\cdots, j-1; \; j = 2,3,\cdots,N)$,
2. $\sum_{i\varepsilon S} \sum_{j\varepsilon \overline{S}} x_{ij} \geq 2$ for all nonempty partitions $(S, \overline{S})$ such that if $(S,\overline{S})$ is considered, then $(\overline{S}, S)$ is not.

$x_{ij} = 0$ if the link (i,j) is not in the tour, and

$x_{ij} = 1$ if the link (i,j) is in the tour.

The disadvantage in finding an optimal tour by integer programming is that it requires many variables and many inequalities. Hence, again, the program is only suitable for small N. Several modifications to linear programming have been given with fewer variables and inequalities. Martin [7] claims to have solved a 42-location problem. However, other articles, [2] and [8], have reported discouraging results with integer programming.

All four of the previous algorithms are exact procedures, and since they are inadequate for a problem with a large number of locations, methods have been devised to give solutions which compare favorably to the exact solution. Several of these methods are iterative in that they improve initial tours.

5. Partitioning

Held and Karp [2], used partitioning with dynamic programming. This is an iterative procedure, which uses an initial tour. The initial tour is partitioned into u ordered sets, each consisting of locations which occur successively in the initial tour. By treating each partition as a location, a u-location traveling salesman problem is created. If $(i_1, i_2, \cdots, i_p)$ and $(j_1, j_2, \cdots, j_q)$ are two ordered sets of a partition, then the distance between the two ordered sets is $a_{i_p j_1}$. If u is not too large, then the u-location problem can be solved by an exact scheme. The solution will have placed each ordered set into the best position, which will have equaled or improved the initial route. In essence, the ordered sets are moved about to produce a better solution.

Different partitions may be used to produce different solutions. Held and Karp defined two types of partitioning, local and global. In a local partition, each of the ordered sets, except one, consists of a single element. This determines the best tour over a local part of the partition. A global partition takes each ordered sets nearly equal in size.

Held and Karp used dynamic programming to solve the sub-traveling salesman problem. They presented several good results on locations of size 42, 20, 48, and 36.

6. R - optimal

Another iterative scheme which uses an initial tour is r-optimal. Lin [9] defines a tour to be r-optimal if it is impossible to obtain a tour with smaller distance by replacing any r of its links by any other set of r links. Figure 5 illustrates 3 links being removed in an 8 location problem. The three removable links are a, b, and c. Two of the routes which can then be formed are 1, 2, 3, 6, 4, 5, 8, 7, 1; and 1, 2, 3, 8, 5, 4, 6, 7, 1.

Lin discovered from experimenting that r = 3 gives excellent results with small computation time compared to r = 4. Since all 3-optimal routes are also 2-optimal, he restricted his algorithm to 3-optimal routes.

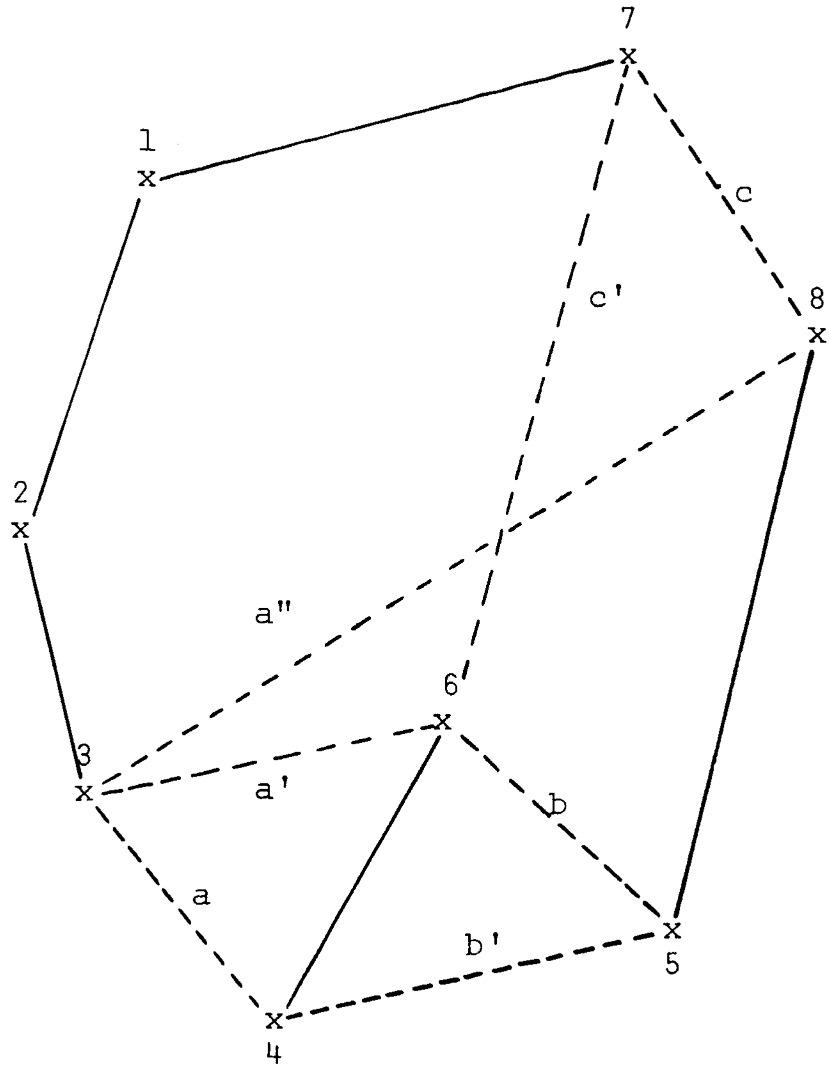

Figure 5. Tour with three removable links

Lin proved a theorem which he used as a basis for his algorithm in producing 3-optimal tours. It is as follows: A tour is 3-optimal if and only if no section of k consecutive locations in the tour can be removed and reinserted (as is, or inverted) between any two consecutive remaining locations to produce a tour with less total distance.

He modified the 3-optimal procedure in two ways. First, he started with an initial tour and then proceeded to find a 3-optimal tour by successively placing k consecutive locations ($k = 1, 2, \cdots, N$) between two other consecutive locations. As soon as he found an improvement, he took this new tour and started the procedure over again, after first rotating the locations to the next consecutive locations. The algorithm stops if no improvement can be made by placing k consecutive locations between any two other consecutive locations.

This program took a relatively short time on a computer, so Lin modified it a second way by calculating m 3-optimal solutions from m random initial tours. The links that were common to all m 3-optimal tours were then removed, with the premise that any link common to all m 3-optimal tours will also be a link in an optimal tour. This reduces the number of locations, and hence, the size of the problem. The procedure is then repeated. (Lin does not say what he would do if there were no common links).

Bellmore and Nemhauser [4] stated in their summary of the traveling salesman problem that they would use dynamic programming if the number of locations were less than or equal to 13. For symmetric problems up to 40 locations, they recommend the branch and bound algorithm. Then for problems which can not be solved by exact schemes, they suggest Lin's 3-optimal algorithm.

B. VEHICLE DISPATCH PROBLEM

The vehicle dispatch problem is a generalization of the traveling salesman problem. The difference between the two is that the vehicle dispatch problem may use more than one salesman, and may also have restrictions on the distance that each salesman may travel. The problem may also be applied to a fleet of trucks which must deliver products to various locations when there are restrictions on the number of miles traveled by a single truck and a load capacity for a single truck.

A mathematical formulation of the problem is as follows:

Given;

1. A set of N locations including the depot,
2. A distance matrix $A = (a_{ij})$ which specifies the distance between location i and location j,
3. A demand vector $Q = (q_i)$ which specifies the demand at location i,
4. The truck capacity C,
5. The maximum mileage L that a truck may travel.

To determine; M routes $(i_{11}, i_{12}, \cdots, i_{1k_1}; i_{21}, i_{22}, \cdots, i_{2k_2}; \cdots; i_{M1}, i_{M2}, \cdots, i_{Mk_M})$ such that

1. $\sum_{p=1}^{k_j} q_{i_{jp}} \le C$ for $j = 1, 2, \cdots, M$ (load constraint),

2. $D_j = \sum_{p=1}^{k_j-1} a_{i_{jp} i_{jp+1}} + a_{i_{jk_j} i_{j1}} \leq L,$

(distance constraint)

so that $\sum_{j=1}^{M} D_j$ (total distance) is a minimum, where

i_{j1} is the depot, and every location is visited once and only once.

1. Conversion into Traveling Salesman Problem

Christofides and Eilon [10] presented a procedure to transform the vehicle dispatch problem into a traveling salesman problem with certain constraints. They augmented the distance matrix with M artificial depots. All of them have the same location, with infinity assigned to the distance between two depots to prevent traveling from one depot to another. This is illustrated in figure 6.

The number of artificial depots augmenting the distance matrix will begin with a small number and increase by one until there exists a feasible solution. Christofides and Eilon suggest the lower bound of $[\sum_{i=1}^{N} q_i/_C] + 1$ for the first value of M since $M \cdot C \geq \sum_{i=1}^{N} q_i$.

In building the route, using a traveling salesman algorithm, a check must be made after each location is added to the route, to see if the distance constraint or the load capacity is violated between two artificial depots.

$$
\begin{array}{ll}
 & \overbrace{\begin{array}{cccc} \hphantom{a_{11}} & \hphantom{a_{11}} & \hphantom{\cdots} & \hphantom{a_{11}} \end{array}}^{M} \\
M\left\{\begin{array}{c} \\ \\ \\ \\ \end{array}\right. &
\begin{array}{cccc|cccc}
\infty & \infty & \cdots & \infty & a_{11} & a_{12} & \cdots & a_{1N} \\
\infty & \infty & \cdots & \infty & a_{11} & a_{12} & \cdots & a_{1N} \\
\vdots & \vdots & & \vdots & \vdots & \vdots & & \vdots \\
\infty & \infty & \cdots & \infty & a_{11} & a_{12} & \cdots & a_{1N} \\
\end{array} \\
 &
\begin{array}{cccc|cccc}
a_{11} & a_{11} & \cdots & a_{11} & a_{11} & a_{12} & \cdots & a_{1N} \\
a_{21} & a_{21} & \cdots & a_{21} & a_{21} & a_{22} & \cdots & a_{2N} \\
\vdots & \vdots & & \vdots & \vdots & \vdots & & \vdots \\
a_{N1} & a_{N1} & \cdots & a_{N1} & a_{N1} & a_{N2} & \cdots & a_{NN}
\end{array}
\end{array}
$$

Figure 6. Augmented distance matrix

Christofides and Eilon used the branch and bound algorithm by Little et al. [6] on several problems. They concluded that this method was inadequate because it could not solve a problem with more than 20 locations, since both the computation time and memory - space requirements became exhaustive. Bellmore and Nemhauser [4]. have reported solutions to problems of size 40.

This procedure is an exact algorithm in that it yields an optimal solution. It is the only exact algorithm for the vehicle dispatch problem known to the author. Vehicle dispatch problems that have a practical application are too large to be solved by means of any known exact algorithm. Hence,heuristic procedures have been developed to handle these large problems.

2. Savings Approach

The vehicle dispatch problem was first presented by Dantzig and Ramser [11] in 1959. The method they employed to solve the problem has become known as the savings approach. Clark and Wright [8] modified the method in an article in 1964 and restricted the load capacity to the same quantity for each vehicle. Gaskell [12] and Christofides and Eilon [10] also gave modifications of the savings approach.

The algorithm begins by linking each location with one vehicle and then returning to the depot. Links are then joined to eliminate vehicles by means of the "savings" equation, $s_{ij} = a_{1i} + a_{1j} - a_{ij}$. This quantity represents the amount saved by joining location i to location j. (See figure 7.) The depot is represented by 1. The total distance for the two vehicles before they are joined is $2a_{1j} + 2a_{1i}$. After the two locations are joined for one route, the total distance is $a_{1j} + a_{ij} + a_{1i}$. Hence the savings is $2a_{1j} + 2a_{1i} - (a_{1j} + a_{ij} + a_{1i}) = a_{1i} + a_{1j} - a_{ij}$. The largest savings, s_{ij}, is selected and checked to see if the constraints are satisfied after locations i and j are joined. If the link i to j is feasible, it is added. Otherwise another link is considered. The solution is obtained when no more links can be added.

Tillman and Cochran [13] further revise Wright and Clark's algorithm by checking the next largest savings after the pair of points is joined. The sum of the two savings is calculated. The second step proceeds as the first, only taking the second largest savings first. A new sum is then calculated. The above procedure is repeated for the third highest savings, the fourth highest savings, etc., until all feasible savings have been included. The largest sum, after linking the pair of points, is then used.

3. R-optimal

Christofides and Eilon [10] introduce the implementing

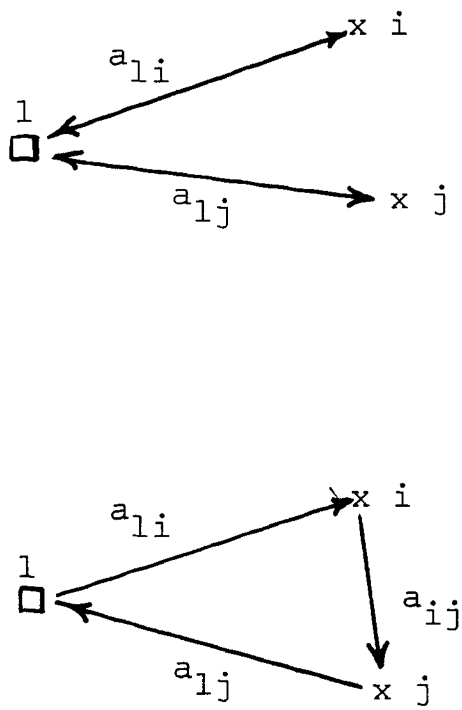

Figure 7. Tour before and after joining two locations

of the 3-optimal algorithm to the dispatch problem. They add artificial depots similar to the way the depots are added in the method described in section A. After generating a random tour, they find a 2-optimal tour. This tour is then used as an initial tour for the 3-optimal algorithm. The constraints are checked after a better tour has been generated. The distance and demand constraints are checked between each two successive depots. If the constraints are satisfied, then three more links are changed until a better tour cannot be formed by changing three links.

It appears from Christofides and Eilon's algorithm that they do not use Lin's algorithm [9] to generate a 3-optimal tour. Rather, they use Lin's definition of a 3-optimal tour: A 3-optimal tour is a tour that cannot be improved by removing 3 links and replacing them by 3 other links. However, they do use the method proposed by Lin, which starts a new search for the 3-optimal tour as soon as a better tour is determined.

The conclusion reached by Christofides and Eilon is that the 3-optimal algorithm gives better routes than the savings method. Christofides and Eilon did not include the distance constraint in the 3-optimal algorithm.

4. Hayes' Algorithm

Hayes [14] developed a heuristic approach for the vehicle dispatch problem in much the same way that a dispatcher would dispatch his fleet of trucks. He first estimates the number of routes that he will need and then picks the same number of outside points. The first outside point is the point furthest from the depot. The other outside points are those obtained by maximizing the quantity, $a_{i1} \cdot \prod_{k=2}^{r} a_{j_k i}$, over all locations i that are not already outside points, and where j_k are outside points and 1 is the depot. This is illustrated in figure 8. The algorithm then chooses one outside point, and adds locations to this point until a tour has reached either a distance or a demand restriction. Then a new outside point is chosen and the remaining points serve as candidates for the next tour. The points are added to the tour according to a score which is assigned to each point. This score is comprised of a variety of different values;

a) Its demand, Q_i,

b) Its distance from the depot, a_{i1},

c) Its distance from the line joining the outside point and the depot, d_i,

d) Its distance to the nearest unassigned point, f_i.

These values are shown in figure 9, where x represents the assigned points, o represents the outside point, and z represents the unassigned points.

The score for a location is a linear combination of these quantities, with the coefficients of Q_i, a_{i1}, and f_i being positive and the coefficient of d_i negative;

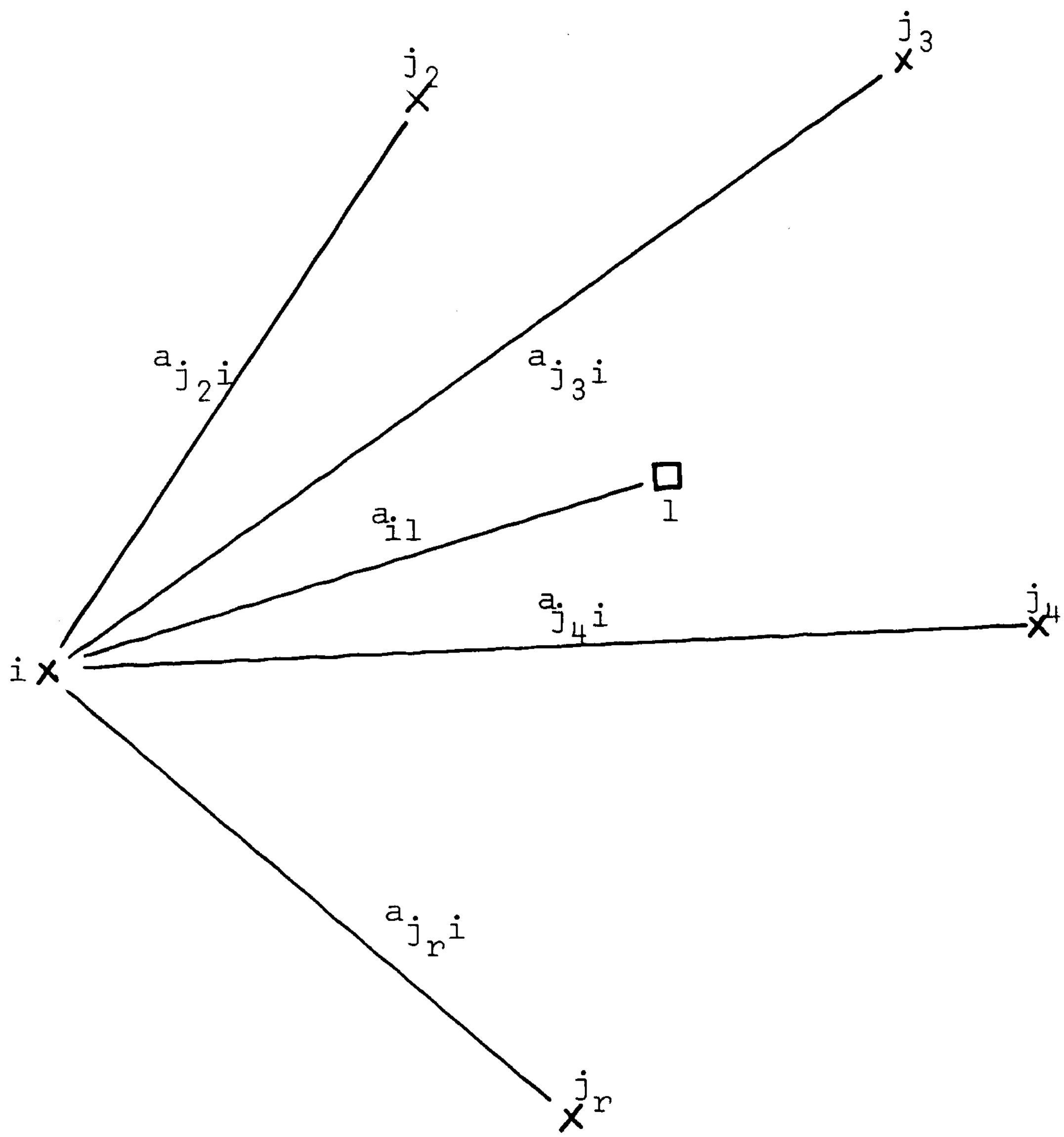

Figure 8. Determining outside points

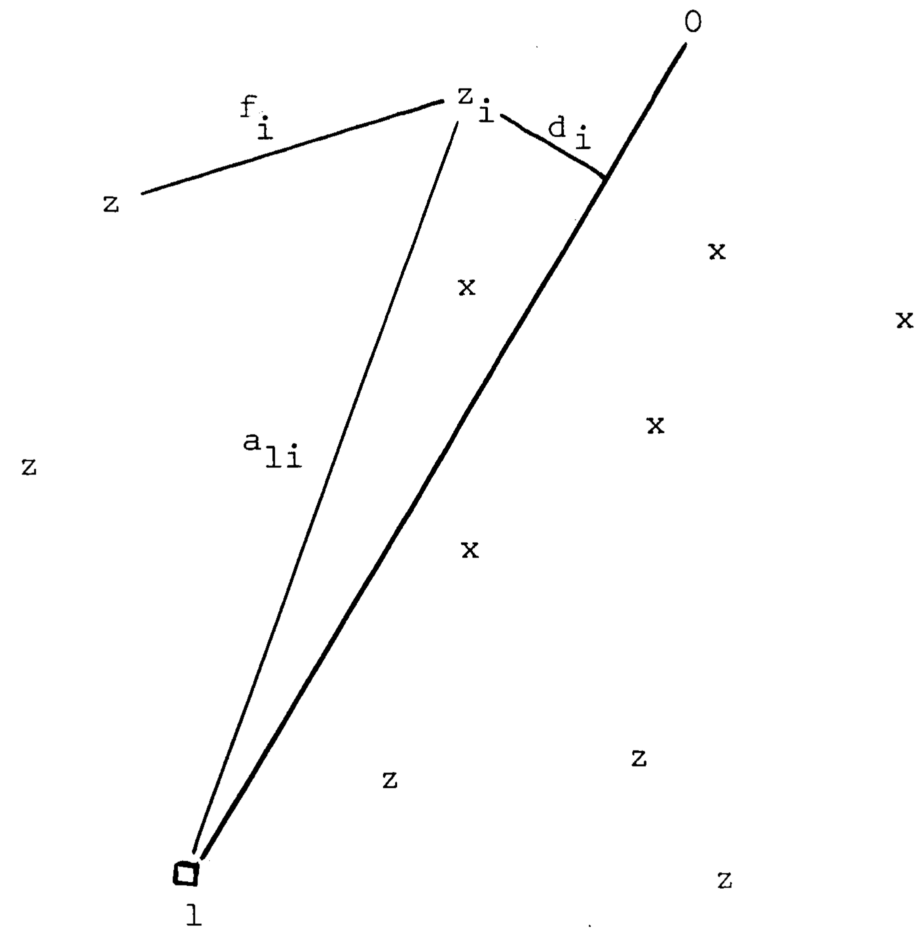

Figure 9. Determining score for point z_i

a point will be added to a route if Q_i, a_{i1}, and f_i are large and d_i is small. Hayes does not advocate any particular values for the coefficients. However, he did run some tests on several problems. The location with the largest score is selected and then the constraints are checked. If the constraints are satisfied, then the location is added to the route. If one of the constraints is not satisfied, the location with the next largest score is examined. After two attempts to add a location, the route is closed and a new outside point is chosen. The algorithm is complete when all locations are assigned. If more outside points are needed, then the unassigned location whose distance from the depot is the greatest, is assigned as an outside point.

It is difficult to compare Hayes' method with other algorithms, since he did not give any results using the method. Also the method used to obtain the values of the coefficients is vague.

III. VEHICLE DISPATCH PROBLEM

A. MATHEMATICAL DEVELOPMENT OF SWEEP ALGORITHM

The purpose of this chapter is to present the mathematical foundations for the vehicle dispatch problem and the sweep algorithm.

DEFINITION 1. The vehicle dispatch location problem (VDLP) is a set of integers, $S = \{1, 2, \cdots, N\}$, containing at least two elements; two positive real numbers, C and D; and the following functions:

a) $Q(I)$, a positive real valued function defined on S', where $S' = S - \{1\}$,

b) $A(I,J)$, a real valued function defined on $S \times S$,

c) $X(I)$ and $Y(I)$, two real valued functions defined on S,

which satisfy the following constraints:

d) $Q(I) \leq C$ for all $I \in S'$,

e) $A(I,J) > 0$ for all I and $J \in S$ except $I = J$,

f) $A(1,I) + A(I,1) \leq D$ for all $I \in S'$,

g) $A(I,I) = 0$ for all $I \in S$.

In the vehicle dispatch problem the set S represents the N locations with 1 as the depot. $Q(I)$ represents the demand for location I, and $A(I,J)$ represents the distance between locations I and J. $X(I)$ and $Y(I)$ are the rectangular coordinates for location I. C and D represent the load and distance capacities respectively for each vehicle.

DEFINITION 2. $SUM(P) = \sum_{I \varepsilon P} Q(I)$ for all $P \subset S'$.

DEFINITION 3. $DIST(P) = \min_{\alpha \varepsilon Per(P)} [A(1, \alpha(1)) + \sum_{i=2}^{n(P)} \overline{A(\alpha(i-1), \alpha(i))} + A(\alpha(n(P)),1]$ for all $P \subset S'$ where $Per(P)$ is the set of all permutations of elements of P and $n(P)$ is the cardinality of P.

DIST(P) is the minimum distance for traveling through all locations in P, starting and ending at 1.

DEFINITION 4. $An(I) = \arctan((Y(I) - Y(1))/(X(I)-X(1))$ where $-\pi < An(I) < 0$ if $Y(I) - Y(1) < 0$, and $0 \leq An(I) \leq \pi$ if $Y(I) - Y(1) \geq 0$, for all $I \varepsilon S'$.

Let us assume that the locations (elements of S') are arranged so that $An(I) < An(I+1)$. (If there exists an I and a J such that $An(I) = An(J)$ then $I<J$ if $A(1,I)<A(1,J)$. This determines a unique ordering).

DEFINITION 5. A P-sect is a nonempty set, P, of elements such that

a) $P \subset S'$,

b) If $I \varepsilon P$, $J \varepsilon P$, $K \varepsilon S'$, and $I<K<J$, then $K \varepsilon P$,

c) If $N \notin P$ then either $SUM(P \cup \{L+1\}) > C$ or $DIST(P \cup \{L+1\}) > D$ where L is the L.U.B. for P,

d) $SUM(P) \leq C$,

d) $DIST(P) \leq D$.

DEFINITION 6. E is a dispersement if and only if $E = \{P_1, P_2, \cdots, P_k\}$ such that

a) $P_i \cap P_j = \emptyset$ for all $i = 1, 2, \cdots, k$; $j = 1,2,\cdots,k$; and $i \neq j$,

b) $\bigcup_{i=1,\cdots,k} P_i = S'$.

DEFINITION 7. $E = \{P_1, P_2, \cdots, P_k\}$ is a P-sect dispersement if and only if P_i is a P-sect for $i = 1,2,\cdots, k$, and E is a dispersement.

DEFINITION 8. The P-sect P_2 follows P_1 if and only if there exists an $I \in P_1$ such that $I + 1 \in P_2$, where $P_1 \subset S'$.

THEOREM 1. A P-sect dispersement for a VDLP exists and is unique.

PROOF. To prove existence, we construct a dispersement whose elements are P-sects. Let $P_1 = \{2,3,\cdots, I_1\} \subset S'$ such that conditions c,d, and e of definition 5 are satisfied. $P_1 \neq \emptyset$ since $2 \in S'$ by definition of VDLP statements d and e. Therefore P_1 is a P-sect. If $I_1 = N$, then the theorem is complete in that there is only one set in the dispersement. If $I_1 < N$, then let $P_2 = \{I_1+1, I_1+2, \cdots, I_2\} \subset S'$ such that conditions c,d, and e of definition 5 are satisfied. Hence P_2 is a P-sect. Likewise define $P_3, P_4, \cdots, P_k$ until P_k contains N. $E = \{P_1, P_2, \cdots, P_k\}$ is a P-sect dispersement since each P_i is a P-sect, the P_i's are disjoint, and every element in S' is in a P-sect.

To show uniqueness it is sufficient to show that the construction of the P_i's is unique. Since $2 \in S'$ it must be in one of the P_i's. P_1 was constructed so as to contain 2. Now P_1 cannot contain any more elements by definition 5 statements d and e, and it cannot contain any fewer elements by definition 5 statement c. Hence P_1 is uniquely determined. Likewise $P_2, P_3, \cdots, P_k$ are uniquely determined. Therefore the construction of each P_i is unique and hence the P-sect dispersement for a VDLP is unique.//

DEFINITION 9. Let $\theta = \{P_1, P_2, \cdots, P_k\}$ be a dispersement. The total distance of θ, $(TD(\theta))$, is defined to be $TD(\theta) = \sum_{i=1}^{k} DIST(P_i)$.

DEFINITION 10. A dispersement, $\theta = \{P_1, P_2, \cdots, P_k\}$ is an optimal dispersement if and only if $TD(\theta) \leq TD(R)$ where R is any dispersement.

There exists a VDLP such that the P-sect dispersement is not an optimal dispersement. This can easily be verified by the following example:

Let $S = \{1,2,3,4,5\}$, $C = 2$, and $D = 15$. Let the functions X(I), Y(I), Q(I) and A(1,J) be defined according to figure 10. By examining the functions it is easy to verify that conditions d,e,f, and g of definition 1 are satisfied. Hence, it is a VDLP. Examination of all the X(I) and Y(I) values reveals that $An(M) \leq An(M+1)$, for M = 1,2,3 and 4.

Let $P_1 = \{2,3\}$ and $P_2 = \{4,5\}$. $\theta = \{P_1, P_2\}$ is a dispersement and the total distance for θ is:

$TD(\theta) = DIST(P_1) + DIST(P_2) = (1+4+5) + (6+4+3) = 23.$

θ is the P-sect dispersement.

Another dispersement is $\theta_1 = \{T_1, T_2\}$ where $T_1 = \{2,5\}$ and $T_2 = \{3,4\}$. The total distance for θ_1 is:

$TD(\theta_1) = DIST(T_1) + DIST(T_2) = (1+2+3) + (5+1+4) = 16.$

Since $TD(\theta) > TD(\theta_1)$, the P-sect dispersement is not an

A =

I	1	2	3	4	5
1	0	1	5	6	3
2	1	0	4	5	2
3	5	4	0	1	5
4	6	5	1	0	4
5	3	2	5	4	0

I	1	2	3	4	5
X(I)	0	1	5	5	1
Y(I)	0	0	1	2	2
Q(I)	1	1	1	1	1

Figure 10. Examples for A,X,Y, and Q

optimal dispersement.//

In an application involving the vehicle dispatch location problem we desire to find an algorithm which will produce an optimal dispersement. However, for large N this becomes exceedingly difficult. Hence, we are satisfied with a dispersement with total distance close to the total distance of an optimal dispersement. Since a P-sect dispersement is not necessarily an optimal dispersement a P-sect dispersement is changed so as to minimize the total distance. This leads to the definition of a modified P-sect.

DEFINITION 11. Let $\{P_1, P_2, \cdots, P_k\}$ be a dispersement and let P_{i+1} be a P-sect. P_i' is a modified P-sect of P_i if and only if $P_i' = (P_i \cup \{M\}) - \{K\}$ where M is the Q_2 and K the Q_1 such that $H = DIST((P_i \cup \{Q_2\}) - \{Q_1\}) + DIST((P_{i+1} \cup \{Q_1\}) - \{Q_2\})$ is a minimum for all $Q_1 \varepsilon P_i$ and $Q_2 \varepsilon P_{i+1}$ and with $DIST(P_i') \leq D$, $SUM(P_i') \leq D$ where P_{i+1} is the P-sect that follows P_i. If $H \geq DIST(P_i) + DIST(P_{i+1})$ then $P_i' = P_i$.

THEOREM 2. In definition 11, $\{P_1, P_2, \cdots, P_{i-1}, P_i', T_{i+1}, P_{i+2}, \cdots, P_k\}$ is a dispersement where $T_{i+1} = (P_{i+1} \cup \{K\}) - \{M\}$ if $P_i' \neq P_i$ otherwise $T_{i+1} = P_{i+1}$.

PROOF: If $P_i' = P_i$, then $\{P_1, \cdots, P_i', T_{i+1}, \cdots, P_k\}$ is a dispersement since $\{P_1, \cdots, P_k\}$ is a dispersement.

If $P_i' \neq P_i$, then each of the sets of $\{P_1, \cdots P_i', T_{i+1}, \cdots, P_k\}$ are disjoint and every element in S' is in at least one of the sets, since just two elements of the two sets were interchanged, and since $\{P_1, P_2,\cdots,P_k\}$ is a dispersement. Hence, the set $\{P_1,\cdots,P_i',T_{i+1},\cdots,P_k\}$ is a dispersement.//

By modifying one set at a time, beginning with P_1, a dispersement can be completely modified. Let $\{P_1', T_2, P_3,\cdots, P_k\}$ be the dispersement with P_1' the modified P-sect of P_1, and $T_2=(P_2-\{M\})\cup\{K\}$ if $P_1\neq P_1'$ and $T_2=P_2$ otherwise. Then let P_2' be the modified P-sect of T_2 with $T_3 = (P_3- \{M\})\cup\{K\}$ if $T_2 \neq P_2'$ and $T_3 = P_3$ otherwise. Continuing this process through P_{k-1} the dispersement $\{P_1',\cdots,P_{k-1}',T_k\}$ is obtained. This dispersement is called the modified dispersement of the P-sect dispersement $\{P_1,P_2,\cdots,P_k\}$.

THEOREM 3. $TD(\{P_1', P_2',\cdots, P_{k-1}', T_k\})\leq TD(\{P_1, \cdots, P_k\})$

PROOF. By definition 11 a P-sect is only changed if the sum of the two DIST values of each set is decreased. Hence, by the definition of the total distance of a dispersement, the total distance of the modified dispersement is less than or equal to the total distance of the P-sect dispersement.//

There may exist a dispersement for which the total distance can be improved by exchanging one location in P with two locations in the P-sect which follows P. This leads to the definition of a second modified P-sect.

DEFINITION 12. Let $\{P_1,P_2,\cdots,P_k\}$ be a dispersement and P_{i+1} a P-sect that follows P_i. Then P_i'' is called a second modified P-sect of P_i if and only if $P_i'' = (P_i \cup \{M\} \cup \{L\}) - \{K\}$ where M is the Q_2, L the Q_3 and K the Q_1 such that

$H = DIST((P_i \cup \{Q_2\} \cup \{Q_3\}) - \{Q_1\}) + DIST(((P_{i+1} \cup \{Q_1\}) - \{Q_2\}) - \{Q_3\})$

is a minimum for all $Q_1 \varepsilon P_i$, $Q_2 \varepsilon P_{i+1}$, and $Q_3 \varepsilon P_{i+1}$ and with $DIST(P_i') \leq D$, $SUM(P_i') \leq D$.

If $H \geq DIST(P_i) + DIST(P_{i+1})$, then $P_i'' = P_i$.

THEOREM 4. In definition 12, $\{P_1, P_2, \cdots, P_{i-1}, P_i'', W_{i+1}, P_{i+2}, \cdots, P_k\}$ is a dispersement where $W_{i+1} = ((P_{i+1} \cup \{K\}) - \{M\}) - \{L\}$ if $P_i'' \neq P_i$, otherwise $W_{i+1} = P_{i+1}$.

PROOF. If $P_i'' = P_i$, then $\{P_1, \cdots, P_i'', W_{i+1}, \cdots, P_k\}$ is a dispersement since $\{P_1, P_2, \cdots, P_k\}$ is a dispersement.

If $P_i'' \neq P_i$, then each of the sets of $\{P_1, \cdots, P_i'', W_{i+1}, \cdots, P_k\}$ are disjoint and every element in S' is in at least one of the sets, since just two elements of P_i were exchanged for one element of P_{i+1} and since the set $\{P_1, P_2, \cdots, P_k\}$ is a dispersement. Hence, $\{P_1, \cdots, P_i'', W_{i+1}, \cdots, P_k\}$ is a dispersement.//

By determining the second modified P-sect beginning with P_1, it is possible to completely determine the second modification of a dispersement. Let $\{P_1'', W_2, P_3, \cdots, P_k\}$ be the dispersement with P_1'' the second modified P-sect of P_1 and $W_2 = ((P_2 \cup \{K\}) - \{M\}) - \{L\}$ if $P_1 = P_1''$ and $W_2 = P_2$ otherwise. Then let P_2'' be the second modified P-sect of W_2 with $W_3=((P_3 \cup \{K\})-\{M\})-\{L\}$ if $W_2 = P_2''$ and $W_3 = P_3$ otherwise. Continuing this through P_{k-1} we obtain the following dispersement: $\{P_1'',P_2'',\cdots,P_{k-1}'', W_k\}$. This dispersement is called the second modified dispersement of the P-sect dispersement $\{P_1,P_2,\cdots,P_k\}$.

THEOREM 5. $TD\{P_1'', P_2'',\cdots,P_{k-1}'', W_k\}) \leq TD(\{P_1', P_2',\cdots,P_{k-1}', T_k\})$ where $\{P_1'',P_2'',\cdots,P_{k-1}'',W_k\}$ is the second modified dispersement and $\{P_1',P_2', \cdots, P_{k-1}', W_k\}$ is the modified dispersement of the P-sect dispersement $\{P_1,P_2,\cdots, P_k\}$.

PROOF. First note in definition 12 that Q_2 may equal Q_3. Hence, K may equal L. But this implies that all possibilities of switching one location of P_i with one location in P_{i+1} are considered. This however, is the definition for a modified P-sect. Hence, $DIST(P_i') + DIST(T_{i+1}) \geq DIST(P_i'') + DIST(W_{i+1})$. Therefore, the total distance of a second modified dispersement is less than or equal to the total distance of a modified dispersement.//

COROLLARY 1. $TD(\{P_1'',P_2'',\cdots, P_{k-1}'', W_k\}) \leq TD(\{P_1,P_2,\cdots,P_k\})$.

PROOF. This follows immediately from theorems 3 and 5.//

Examples of a vehicle dispatch location problem can be given for which the second modified dispersement is not an optimal dispersement.

B. SWEEP ALGORITHM PROCEDURE

The mathematical development in the previous section provides the basis for the sweep algorithm. The locations are partitioned into a P-sect dispersement and then into a second modified P-sect dispersement. Corollary 1 assures us that a second modified P-sect dispersement has a total distance less than or equal to the total distance of a P-sect dispersement. A second modified P-sect dispersement may be obtained by rotation the X and Y axes counterclockwise so that the first location will become the last location, the second location will become the first location and so forth. This process of rotating the X and Y axes is continued until a new P-sect dispersement cannot be generated. Each time, the total distance of the second modified P-sect dispersement is calculated. The minimum of these total distances provides a good heuristic solution for the vehicle dispatch problem.

The algorithm begins with location 2 and then adds locations 3,4,... to the route. Recall that the locations were renamed according to the size of the polar coordinate angle; location 1 has the smallest angle; location 2 has the next largest angle, and so forth. This is called the forward procedure. A second method begins with location N and adds locations N-1, N-2,... to the route. This procedure is called the backward procedure. In most cases the two procedures produce different routes.

A disadvantage of the algorithm is that a traveling salesman problem must be solved many times in order to determine a second modified P-sect. This is necessary in order to determine the location which is to be eliminated from the route, and the locations which are to be added to the route. Hence, in the sweep algorithm these locations are determined heuristically by the following procedure: The location to be deleted from the route is obtained by minimizing a function of the radius, R(I), and the angle, An(I), of each location in the route. This provides a location that is both close to the depot and also close to the next route. A function of R and An, which seems to work very well, is R(I) + An(I) · AVR, where AVR is the average of the radii for all locations. For a modified P-sect, the location, I, which is augmented to the route, is the location nearest to the last location that was added to the route. For the second modified P-sect, the other location added to the route is the location nearest to location I. Choosing these locations in this manner may not give the best locations. However, it provides a very fast scheme for selecting the locations, compared to the use of other algorithms, which require solving the traveling salesman problem many times.

If one or two locations are added to the route by this scheme, then the next location is also checked to see if it can be included in the route. This process of adding one or two locations and deleting another location continues until no improvement is found. Hence, an iterative scheme is established. Figure 11 illustrates this scheme with an example of 21 locations and all possible paths between the locations. For example, it is impossible to go directly from location 4 to location 10. Let the distance between two adjacent houses be 1, then A(1,2) = 5, A(2,3) = 1, A(2,6) = 2, A(4,10) = 5, and so forth. Also, let each location have a demand of 1 and let the load capacity be 10. The backward sweep would first assign the locations 21, 20, 19, 18, 17, 16, 15, 14, 13, and 12, since each location is selected according to the value of the angle in polar coordinates. These locations are circled in figure 12. This route has a distance of 18.

The iterative scheme then selects location 11 and deletes location 15, which is shown in figure 13. By applying the iterative scheme two more times, locations 10 and 7 are augmented to the route, while locations 19 and 18 are eliminated. This provides a total distance of 16. This is shown in figure 14.

Another variation involves checking the J + 2 location, where J is the last location added to the route. If the distance and load constraints are satisfied, then the J+2 location is added to the route. This variation will always yield the same number of or fewer routes. However, it may produce a dispersement with greater total distance.

Taking these two variations two at a time gives four possibilities. All four of these possibilities are used in the sweep algorithm.

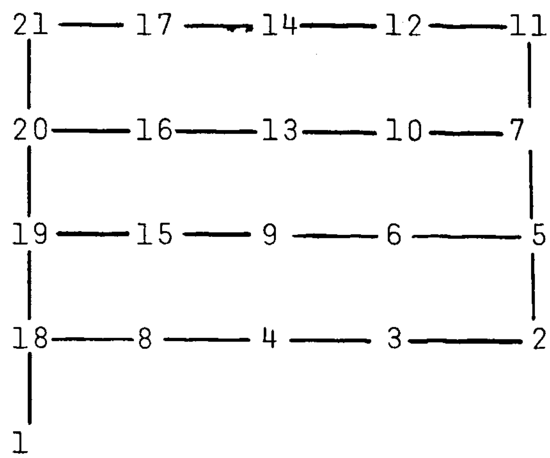

Figure 11. Example of a vehicle dispatch problem

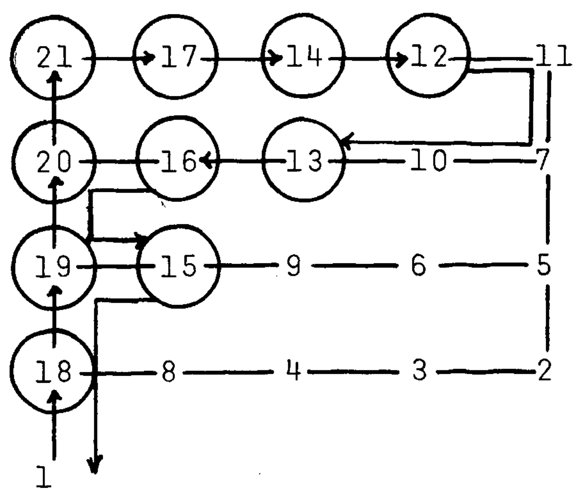

Figure 12. Initial tour

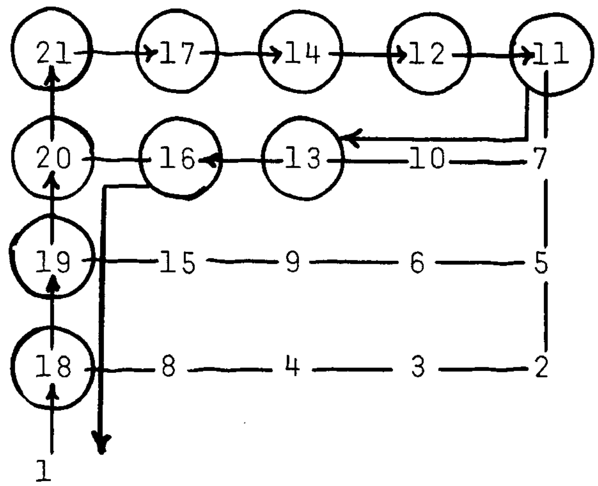

Figure 13. Tour after one iteration

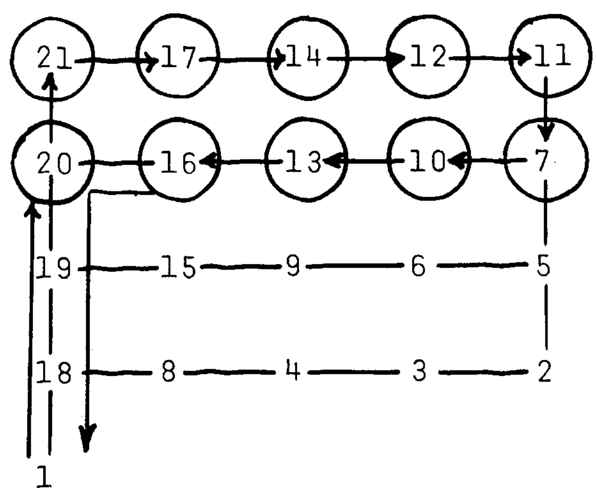

Figure 14. Tour after three iterations

The following step procedure for the sweep algorithm presents the forward procedure and does not check the J + 2 location. We shall assume the notation used in the mathematical development, and also that we have a VDLP. Instead of relabeling the locations, we will let K(I) denote the location with the I^{th} largest angle. Fortran logic is used in explaining the step procedure.

STEP 1.

Evaluate the polar coordinates for each location with the depot at (0,0). Let An(I) represent the angle and R(I) the radius for location I.

STEP 2.

Determine K(I) for I = 1, ···, N such that An(K(I)) is less than or equal to An(K(I+1)).

STEP 3.

Begin the first route with J = 2 and SUM = Q(K(2)).

STEP 4.

Increment the angle by making J = J + 1.

STEP 5.

If SUM + Q(K(J))>C, then go to step 7.

STEP 6.

Augment the route with location K(J) by making SUM = SUM + Q(K(J)). If J = N, then go to step 16. If J ≠ N, then go to step 4.

STEP 7.

Calculate the minimum distance, D_1, for the route, by means of a traveling salesman algorithm. Check the distance constraint. If the distance capacity is exceeded then eliminate K(J-1) from the route. Make SUM=SUM-Q(K(J-1)) and J = J-1. Check the distance constraint again. Continue this procedure until the distance constraint is satisfied.

STEP 8.

Determine JJX so that K(JJX) is the nearest location to K(J-1) and not in a route. Find JII so that K(JII) is the nearest location to K(JJX) and not in a route. Likewise determine I so that R(K(I)) = An(K(I))·AVR is a minimum for all locations in the route. Let KII denote this I. Determine the minimum distance, D_2, for the route with K(JJX) added to the route and K(KII) deleted from the route.

STEP 9.

If $D_2 \leq D$ and the load constraint is satisfied, then go to step 11. Otherwise go to step 10.

STEP 10.

Record the route and start a new route by setting SUM = Q(K(J)). Go to step 4.

STEP 11.

Evaluate the minimum distance, D_3, for starting at 1, traveling through locations K(J), K(J+1),···, K(J+4) and ending at K(J+5). Determine the distance, D_4, for traveling through the same locations, except eliminate K(JJX) and inject K(KII). If K(JJX) is not K(J), K(J+1),···, or K(J+4), then go to step 10. If $D_1 + D_3 < D_2 + D_4$ then go to step 13. Otherwise go to step 12.

STEP 12.

Place K(JJX) in the route and remove location K(KII). Go to step 4.

STEP 13.

Evaluate the minimum distance, D_5, for the route with K(JJX) and K(JII) substituted for K(KII). If K(JJX) and K(JII) are not K(J), K(J+1), $\cdots$, or K(J+4), then go to step 10. If $D_5 < D$ and the load constraint is satisfied then go to step 14. Otherwise go to step 10.

STEP 14.

Determine the minimum distance D_6 for starting at 1; traveling through locations K(J), K(J+1),$\cdots$, K(J+4); and ending at K(J+5), with K(JJX) and K(JII) excluded and K(KII) included. If $D_1 + D_3 < D_5 + D_6$, then go to step 10. Otherwise go to step 15.

STEP 15.

Place K(JJX) and K(JII) in the route and eliminate K(KII) from the route. Go to step 4.

STEP 16.

Evaluate the minimum distance for the route and check the distance constraint. If not satisfied, then go to step 17. If satisfied, then that set of routes is complete. Check to see if another set of routes is needed. If no more are needed, then go to step 19. Otherwise go to step 18.

STEP 17.

Delete one from the route. (J = J - 1.) Go to step 10.

STEP 18.

Increment the angle by one location (i.e. start with K(3) for the second set of routes.) Go to step 2.

STEP 19.

Stop.

Bellmore and Nemhauser have tested several algorithms for the traveling salesman problem and have reported that Lin's 3-optimal did as well, if not better than, other algorithms [4]. Hence, the 3-optimal algorithm was used in the sweep algorithm to determine the sequence of locations which yields a minimum distance for each route.

An algorithm is also needed to determine the minimum distance of traveling through locations K(J), K(J+1),..., K(J+4); starting with 1 and ending at K(J+5). This is not a traveling salesman problem, in that it does not begin or end at the same location. Hence, Lin's 3-optimal algorithm does not apply. In chapter IV, section B, a branch and bound algorithm is used with arbitrary cost functions at each location. This algorithm can easily be modified to determine the minimum distance of a location problem which does not begin and end at the same location. Therefore, it was used in the sweep algorithm.

The sweep algorithm is a heuristic procedure which attempts to minimize the number of servers and the total cost. Contradictory as it may seem, minimizing the number of routes does not necessarily minimize the total cost. This can best be shown by the example given in figure 15. Let

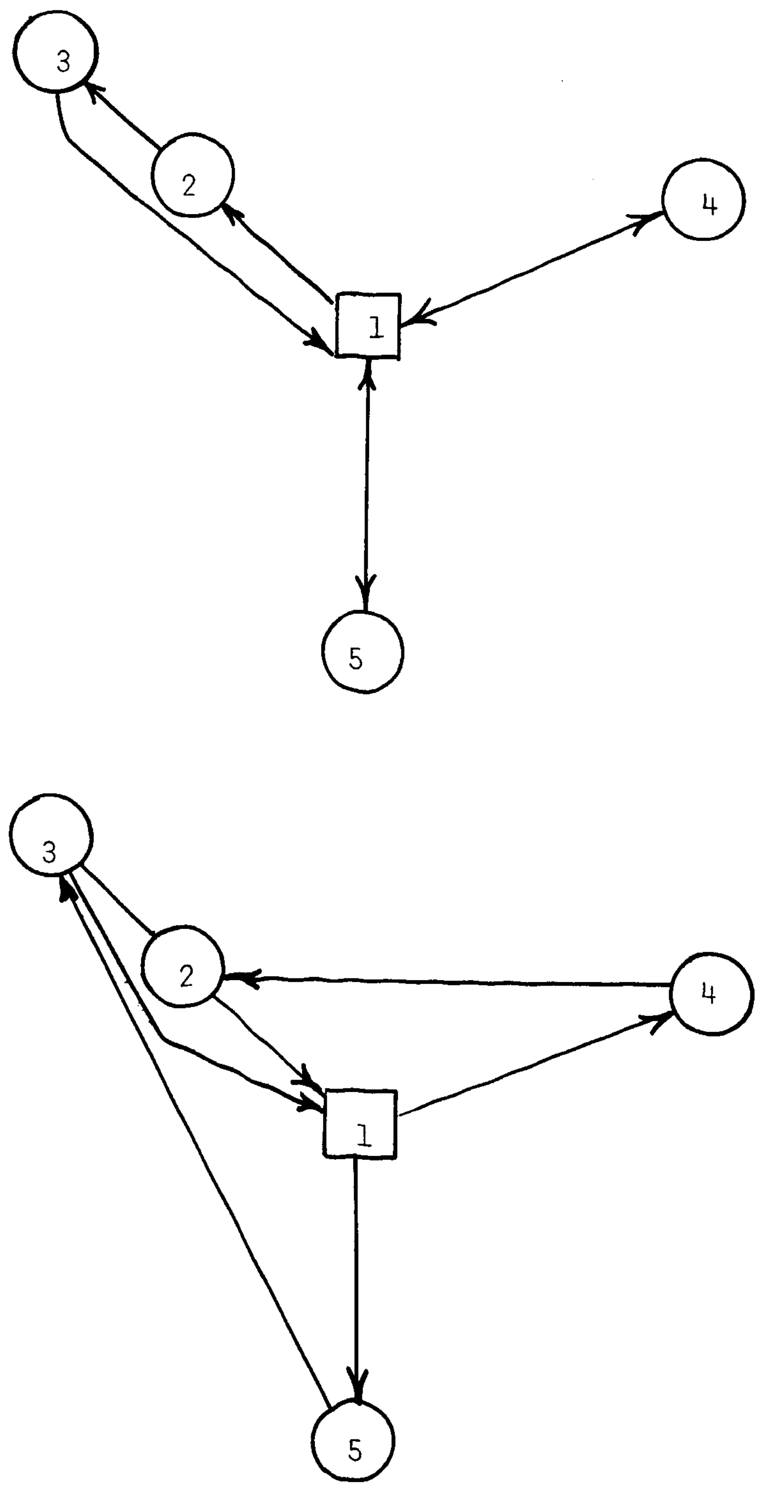

Figure 15. Five locations with two and three routes.

location 1 be the depot at the origin, and the coordinates of the four locations be a follows: location 2 at (-2,2) with demand 20; location 3 at (-4,4) with demand 20; location 4 at (4,2) with demand 40; and location 5 at (0,-5) with demand 40. Let the load limit for the servers be 60. It is possible to construct two routes that will service all four locations, namely routes 1,4,2,1 and 1,5,3,1. This yields a total distance greater than the three routes: 1,2,3,1; 1,4,1; and 1,5,1. Hence, an optimal solution may not have the minimum number of routes.

C. MODIFICATIONS OF SWEEP ALGORITHM

There are various ways in which the sweep algorithm may be modified. Several of these variations were tested and have produced better solutions on particular problems. Four modifications which have been considered are as follows:

1. In the sweep algorithm, the location, K(JJX), which replaces a location already in the route, is the location closest to the last location in the route. Figure 16 shows that this procedure may not yield the best location. Let X represent the locations of a route, Z represent the depot and 0 represent the unassigned locations. Since location 6 is the last location in the route and location 8 is the closest location to 6, the sweep algorithm selects location 8 for K(JJX). However, from figure 16 it is seen that location 7 is a better choice for K(JJX).

Location 7 may be chosen by first requiring R(I) to be large, where I is a location in the route, and then making K(JJX) the M which minimizes A(I,M), where M is an unassigned location. A suggested lower bound for R(I) is AVR·(0.7), where AVR is the average of the R(K)'s for all $K = 2,3,\cdots, N$.

2. Step 8 in section B uses a function of R and Q to determine the location to be deleted from a route. Several functions were used, but none were found to be superior in all cases. The function R(I) + Q(I)·AVR gave better overall results than other functions that were used.

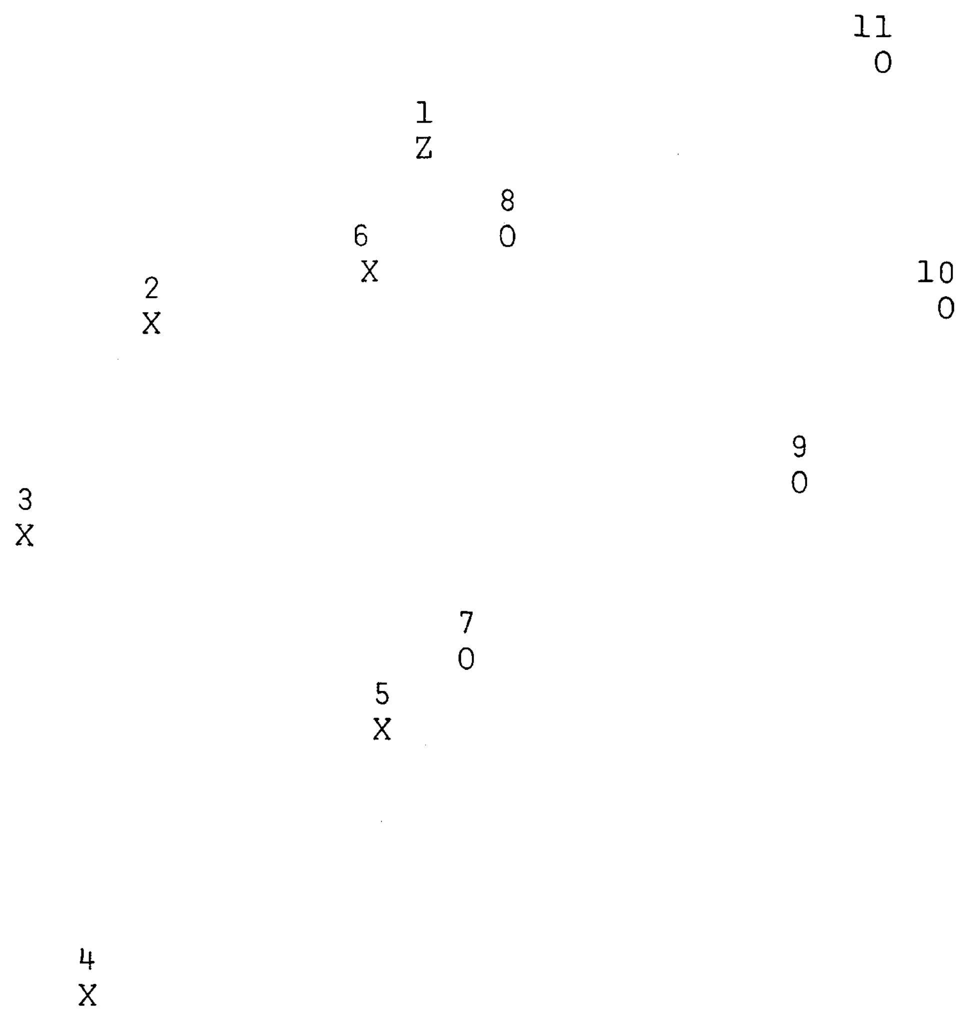

Figure 16. P-sect dispersement and unassigned points

3. Steps 11 and 14 in the sweep algorithm may also be modified to include more locations than K(J+5). This will always provide the same or better solutions. However, as soon as more locations are used, then more time is needed to calculate the minimum distance to traverse the locations.

4. The sweep algorithm examines the second modified P-sect to see if it provided a savings in the total distance. Likewise, a third modified P-sect might also be checked. This involves changing three locations not in the route with one location in the route. Other combinations might also be examined by such means as interchanging two locations for two other locations, or interchanging three locations for two locations. Again the difficulty with checking these possibilities is that it requires many computations since a traveling salesman problem must be solved each time. Examples can be given where these combinations can provide a better solution. However, the problems that were solved did not reveal this. (See appendix B).

IV. GENERALIZED VEHICLE DISPATCH PROBLEM

A. MATHEMATICAL FORMULATION OF THE PROBLEM

The vehicle dispatch problem with arbitrary cost functions is a generalization of the vehicle dispatch problem and is defined as follows:

Given;

1. $\{1,2,\cdots, N\}$, the set which represents N locations,
2. A_{ij}, the time to travel from location i to location j,
3. $f_i(s_i)$, an arbitrary cost function assigned to location i where s_i is the time that location i was serviced by a server,
4. Q_i, the demand for location i,
5. $g(t)$, a cost function which gives the cost to travel for a length of time t.

The problem is to determine the number of routes and the locations for each route so as to minimize the total cost and still satisfy a time and a load constraint for each server.

A mathematical formulation of the problem is as follows: Determine M finite sequences, $p_{i1}, p_{i2}, \cdots, p_{ik_i}$ for

$i = 1,2,\cdots,M$, such that

$$TC = \sum_{i=1}^{M} \{ \sum_{j=2}^{ki} [f_{p_{ij}}(s_{p_{ij}})] + g(s_{p_{ik_i}} + A_{p_{ik_i}p_{i1}})\}$$

is a minimum, where $s_{p_{ij}} = \sum_{q=1}^{j-1} A_{p_{iq}p_{i,q+1}}$, for which the following constraints are satisfied:

1. $\{p_{ij} \mid i = 1,2,\cdots, M \text{ and } j = 2,3,\cdots,k_i\} = \{2,3,\cdots,N\}$,
2. $p_{ij} \neq p_{kr}$ for all i, k, j>1, and r>1 except for i = k when j = r; $p_{i1} = 1$ for all i,
3. $\sum_{j=2}^{k_i-1} (A_{p_{ij}p_{i,j+1}}) + A_{p_{ik_i}p_{i1}} \leq D_i$ for all i,
4. $\sum_{i=2}^{k_i-1} Q_{p_{ij}} \leq C_i$ for all i.

The sequence $p_{i1}, p_{i2},\cdots,p_{ik_i}$ represents the route for server i. The j^{th} location that server i serves is p_{ij}. TC represents the total cost for all M servers, and $s_{p_{ij}}$ is the total time that server i travels through location p_{ij}. The third constraint restricts server i to a time D_i to complete the route. The fourth constraint restricts the total demand for server i to C_i.

B. BRANCH AND BOUND ALGORITHM

Let us assume that the number of servers is one and that there are no constraints. This problem is then a generalization of the traveling salesman problem in that there is a cost function at each location. Mathematically the problem may be stated as follows: Given are A_{ij}, $f_i(s_i)$, $g(t)$ and Q_i. Determine a permutation, $p_2, p_3, \cdots, p_N$, such that

$$TC = g(s_{p_N} + A_{p_N 1}) + \sum_{i=2}^{N} f_{p_i}(s_{p_i})$$

$$\text{where} \quad s_{p_i} = \sum_{q=1}^{i-1} A_{p_q p_{q+1}}$$

is a minimum for all permutations, $p_2, p_3, \cdots, p_N$, of the set, $\{2, 3, \cdots, N\}$.

The branch and bound algorithm can be applied to this problem, but it differs from the algorithm given by Little, et al., in two ways [6]. First, each node represents a location instead of a link, and second, the bound is determined only after a route is completed. These two modifications are necessary since the total time, s_{p_i}, through location p_i is needed in order to obtain the value of the cost function of location p_i. Hence, the algorithm begins at the depot and branches to one of the remaining locations. Figure 17 shows all possible branches for a four-location problem.

After one branch is determined, the total cost for that branch, including returning to the depot, is cal-

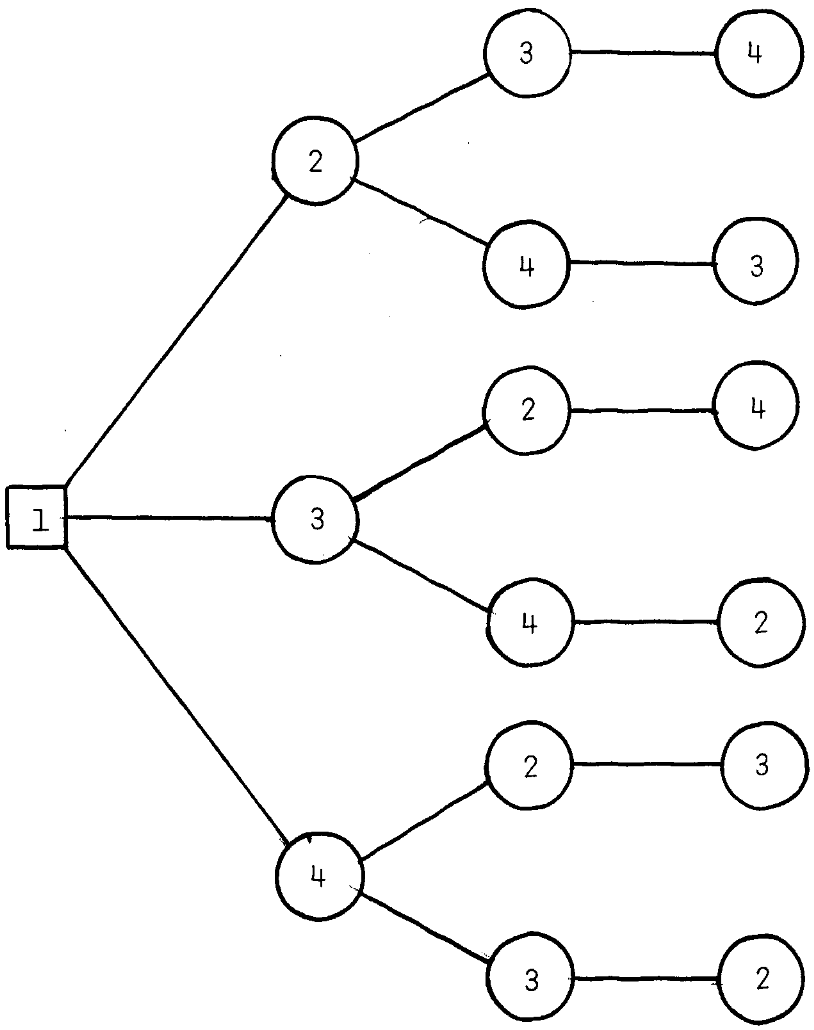

Figure 17. Complete tree for a four-location problem

culated. This cost serves as a bound until another total cost is calculated which is less than the bound. Then this cost becomes the bound. Assuming that $f_i(s_i) \geq 0$ and $s_i \geq 0$, for all i, and $g(t) \geq 0$, for all $t \geq 0$, then a branch can be terminated whenever its total cost exceeds the bound. Putting these restrictions on the functions is not too limiting, since applications will normally have these restrictions.

Another restriction, which in turn aids the algoritm, is to require the functions to be monotonic increasing. This permits the algorithm to back off one node before continuing on a different branch, whenever the total cost exceeds the bound. This is the essence of the following theorem.

THEOREM 6.

If $f_2(s_2), f_3(s_3), \cdots, f_N(s_N)$ and $g(t)$ are monotonic increasing functions, $A_{ij} + A_{jk} \geq A_{ik}$,

$$E = g(s_{p_y}) + \sum_{i=2}^{y} f_i(s_{p_i}) \geq B$$

where B is a positive real number, and $p_2, p_3, \cdots, p_N$ is a permutation of the set $\{2,3,\cdots,N\}$, then

$F = g(s_{r_z}) + \sum_{i=2}^{z} f_i(s_{r_i}) \geq B$ where $r_2, r_3, \cdots, r_N$ is a permutation of the set $\{2,3,\cdots,N\}$ and $r_i = p_i$ for all $i<y$ and $r_z = p_y$. (As before $s_{p_j} = \sum_{i=1}^{j-1} A_{p_i p_{i+1}}$).

PROOF.

Let $k = z - y$. Note that $p_y \varepsilon \{ r_y, r_{y+1}, \cdots, r_N\}$ since

$p_2, p_3, \cdots, p_N$, and $r_2, r_3, \cdots, r_N$ are permutations of the set $\{2,3,\cdots,N\}$ and $p_i = r_i$ for all $i<y$. Hence, $z \geq y$ and consequently $k \geq 0$. The method of proof is mathematical induction on k.

If $k = 0$, then $p_y = r_y$ and hence $E = F$. Then it follows that $F \leq B$ since $E \leq B$.

Assume that the theorem is true for $k = k'$, i.e. $F \leq B$ whenever $r_{k'+y} = p_y$.

Now let $k = k'+1$. Then $z = k'+1+y$ and hence,

$$F' = g(s_{r_{k'+1+y}}) + \sum_{i=2}^{k'+y+1} f_i(s_{r_i}) = g(\sum_{i=1}^{k'+y} A_{r_i r_{i+1}}) + \sum_{i=2}^{k'+y+1} f_i(s_{r_i}).$$

From this it follows that:

$$F' = g(\sum_{i=1}^{k'+y-2} A_{r_i r_{i+1}} + A_{r_{k'+y-2} r_{k'+y-1}} + A_{r_{k'+y-1} r_{k'+y}})$$

$$+ \sum_{i=2}^{k'+y-1} f_i(s_{r_i}) + f_{k'+y}(s_{r_{k'+y}}) + f_{k'+y+1}(s_{r_{k'+y+1}})$$ and also

since $A_{ij} + A_{jk} \geq A_{ik}$ and f_i and g are monotonic increasing functions, we then have;

$$F' \geq g(\sum_{i=1}^{k'+y-2} A_{r_i r_{i+1}} + A_{r_{k'+y-2} r_{k'+y}}) + \sum_{i=2}^{k'+y-1} f_i(s_{r_i}) +$$

$$f_{k'+y+1}(s_{r_{k'+y-1}} + A_{r_{k'+y-1} r_{k'+y+1}}) = G.$$

But G is the value of F for a permutation that has $k=k'$. Hence, its value is less than or equal to B since the theorem is true for $k = k'$. Therefore, $F' \geq B$ and the theorem is proved via mathematical induction.//

The branch and bound algorithm can be modified by using different criteria to select the next location. One method is to choose the location which maximizes the value $f_i(T) - f_i(t)$ where t is the time that the next location will be visited, and T is the time that the last location will be visited. This will increase the possibility for a location with a large increasing cost function to be selected first, while a location with a constant cost function will be selected last. The disadvantage of this procedure is that T is not known until the route is completed. However, estimates such as

$$[((2/3) \sum_{i=1}^{N} \sum_{j>i} A_{ij})/(N^2 - N)] \cdot 2 \cdot (N + 1)$$

can be used for symmetric A matrices.

The following is a step procedure for the branch and bound algorithm for one server and arbitrary cost functions: Let A(I,J) denote the time of travel from location I to location J and f(I,t) denote the cost function for location I at time t.

STEP 1.

Begin the accumulated distance and time, D(1,1) = 0 and T(1,1) = 0. Set I = 1 and Bound equal to large number. Let $Z = [((2/3)\sum_{i=1}^{N} \sum_{j>i} A_{ij})/ (N^2 - N)] \cdot 2 \cdot (N + 1)$.

STEP 2.

Set I = I + 1 and calculate T(I,J) and D(I,J), the total time and cost of the route from depot 1 to location J, where J is the I^{th} route. Do this for all J not already assigned.

STEP 3.

If D(I,J) > Bound, for any unassigned J, go to step 5.

STEP 4.

Select J such that H(I,L) = F(J,Z) - F(J,T(I,J)) is a minimum for all J not assigned. If there are no unassigned J's, then go to step 5. Otherwise go to step 8.

STEP 5.

Set I = I - 1. If I = 1, then go to step 11.

STEP 6.

Find L such that H(I,L) is a minimum for all unassigned L. If there are no unassigned L's, then go to step 5.

STEP 7.

Set IT(I) = L. Go to step 9.

STEP 8.

Set IT(I)= J.

STEP 9.

If I is less than N, then go to step 2.

STEP 10.

Calculate the total time and cost to return to 1 for the route IT(K) for K = 1,2,···,N. If the total cost is less than Bound then set Bound equal to the total cost. Print out the route. Go to step 5.

STEP 11.

Stop.

V. EXPERIMENTS AND RESULTS

The sweep algorithm was used to solve eight vehicle dispatch problems. Appendix B contains the details of these problems. Problems one through four were proposed by Gaskell [12]. All four of these problems have a load and a distance constraint for each server, and an additional distance of ten units for each location. Christofides and Eilon's 3-optimal algorithm [10] does not apply to these problems, since it does not solve problems with distance constraints. The results of Gaskell's savings approach are compared with the four variations of the sweep algorithm in Table I. Problem one has 22 locations, including the depot. All four of the variations of the sweep algorithm were able to schedule all of the locations in 4 routes. Two of these had a total distance that was less than the distance given by the savings approach.

Problem two was the only example in which the algorithm did not provide a smaller total distance than the savings approach. Again all the variations had the same number of routes as the savings approach, namely 5. The best answer of 956 was only 0.5% greater than the solution given by the savings approach.

The sweep algorithm gave better results on problems three and four. In problem four, the sweep algorithm was able to reduce the number of routes from 5 to 4, when the J + 2 location was checked after each route was formed.

Problem Number	Number of Locations	Gaskell's Savings Approach	Christofides and Eilon's 3-optimal	Sweep Algorithm not checking J+2 Forward	Sweep Algorithm not checking J+2 Backward	Sweep Algorithm checking J+2 Forward	Sweep Algorithm checking J+2 Backward	Best Solution
1	22	598 R=4		589 R=4	608 R=4	602 R=4	592 R=4	586 R=4
2	23	949 R=5		969 R=5	956 R=5	962 R=5	995 R=5	956 R=5
3	30	963 R=5		945 R=5	885 R=4	980 R=5	885 R=4	885 R=4
4	33	839 R=5		851 R=5	842 R=5	854 R=5	817 R=4	817 R=4
5	51	585 R=6	556 R=5	574 R=5	553 R=5	575 R=5	546 R=5	524 R=5
6	76	900 R=10	876 R=10	896 R=11	906 R=11	865 R=10	884 R=10	865 R=10
7	101	887 R=8	863 R=8	878 R=8	854 R=8	871 R=8	862 R=8	854 R=8
8	251			5907 R=26	5962 R=26	5794 R=25	5911 R=25	5794 R=25

TABLE I

Comparisons of Vehicle Dispatch Algorithms

Hence, a greater savings was obtained. Problems five, six, and seven were posed by Christofides and Eilon [10]. These problems do not have a distance constraint for the server, nor do they have an additional distance for the locations. At least one of the variations of the sweep algorithm provided a solution which was better than the 3-optimal and the savings approach. In problem six, checking the J + 2 location was necessary to reduce the number of routes from 11 to 10, and consequently produce a smaller total distance.

The real test for a vehicle dispatch algorithm is its ability to solve a problem involving many locations. Problem eight, in appendix B, has 250 locations and this problem was easily solved by the sweep algorithm.

The sweep algorithm modifications presented in chapter III, section C, were also used. Only two improvements were determined out of the eight problems. These were problems one and five. Their results are included under "Best Solution" in Table I and also in Appendix B.

The disadvantage of the sweep algorithm in solving large problems is the time required to solve the traveling salesman problem. If the number of locations for each route remains approximately the same, then the time to solve the vehicle dispatch problem becomes linear with the number of locations. Other algorithms have an exponential growth rather than linear. Hence, the sweep algorithm is capable of solving larger problems.

Problem eight required approximately 15 minutes of computer time, including compiling and execution time on an IBM 360/50. In many cases, the cost for computer time is inexpensive compared to the savings in the total cost that a better route may produce. For example, if the total distance for the routes of a school bus were reduced by 25 miles, then this would provide a larger savings in total cost for one year than the cost for a few minutes of computer time.

The problems solved in the appendix defined distance between two points to be $\sqrt{(X_i - X_j)^2 + (Y_i - Y_j)^2}$. However, the sweep algorithm can be used on other distances. In an application such as the school bus routing, the distance between all locations and the rectangular coordinates for each location must be given. The sweep algorithm uses the same procedure as before, except A(I,J) is now defined according to the actual geographic distance rather than the straight line distance between two locations.

The branch and bound algorithm presented in chapter IV, section B, is an exact scheme. It does have the disadvantage of requiring a large number of calculations for problems with many locations. A ten-location problem with ten cost functions was solved by the algorithm in 10 minutes on an IBM 360/50. The time required to solve a problem depends upon the cost function which determines the lower bound. If a good lower bound is determined on the first route, then more branches can be eliminated, and hence, fewer calculations are required.

VI. SUMMARY, CONCLUSIONS AND FURTHER PROBLEMS

There are many problems that can be classified as vehicle dispatch problems. However, the algorithms presented in chapter II are generally not satisfactory for practical problems, since these problems usually involve many locations. The purpose of this thesis is to develop an algorithm for solving a large problem.

The sweep algorithm is a heuristic procedure for the vehicle dispatch problem. The basic procedure of the algorithm is to aggregate a set of locations into a P-sect dispersement. Then each of these P-sects are examined to see if one or two locations of a P-sect can be switched with one location of another P-sect so as to reduce the total distance. In chapter III, section A, it was shown that a second modified P-sect dispersement has a total distance which is less than or equal to the total distance of a P-sect dispersement. The sweep algorithm heuristically produces a second modified P-sect dispersement. The elements of a modified P-sect are the elements of a route. A traveling salesman algorithm is used to determine the sequence of locations which will yield the least distance in the route. Four sets of routes are developed by the following procedures:

1. Augment the routes by means of the forward procedure and not check the J + 2 location.
2. Augment the routes by means of the forward procedure and check the J + 2 location.

3. Augment the routes by means of the backward procedure and not check the J + 2 location.
4. Augment the routes by means of the backward procedure and check the J + 2 location.

The algorithm is then repeated with the X - Y axes rotated counterclockwise so that the first location is in the last route. The solution given by the sweep algorithm is the dispersement which gives the smallest total distance.

The sweep algorithm was shown to give better solutions than the savings approach in 6 out of 7 problems, and better solutions than the 3-optimal on all 3 problems which Christofides and Eilon proposed. The sweep algorithm was also able to solve a large problem involving 250 locations.

A mathematical formulation of the vehicle dispatch problem with arbitrary cost functions and a branch and bound algorithm which solves the problem for one server were developed. A theorem proved in chapter IV, section B, permits the branch and bound algorithm to solve problems involving 10 locations.

The vehicle dispatch problem may be generalized into several unsolved problems, which also have practical applications. One generalization is a vehicle dispatch problem with more than one depot. This is applicable to the routing of school busses in a school system which has more than one school. Another generalization is the problem to determine the number of depots necessary to minimize the total cost to serve a set of locations. This could be used to determine the number of factories needed to deliver their commodity to a set of stores. Neither of these problems has been solved.

The branch and bound algorithm has the disadvantage of requiring a large number of calculations for large problems. Therefore, heuristic approaches are needed to solve larger problems.

The branch and bound algorithm was also restricted to only one server. Hence, there does not exist an algorithm, exact or heuristic, which will solve the generalized vehicle dispatch problem with arbitrary cost functions and with more than one server.

BIBLIOGRAPHY

1. Flood, Merrill M., "The Traveling Salesman Problem," Operations Research, Vol. 4, 1956, p. 61-75.

2. Held, M. and Carp, R. M., "A Dynamic Programming Approach to Sequencing Problems," J. Soc. Ind. and Appl. Math., Vol. 10, 1962, p. 196-210.

3. Bellman, R., "Dynamic Programming Treatment of the Traveling Salesman Problem," J. Assn. for Computing Machinery, Vol. 9, 1962, p. 61-63.

4. Bellmore, M. and Nemhauser, G. L., "The Traveling Salesman Problem: A Survey," Operations Research, Vol. 16, 1968, p. 538-558.

5. Shapiro, D., "Algorithms for the Solution of the Optimal Cost Traveling Salesman Problem," Sc. D. Thesis, Washington University, St. Louis, 1966.

6. Little, J. D. C., Murty, K. D., Sweeny, D. W., and Karel, C., "An Algorithm for the traveling Salesman Problem," Operations Research, Vol. 11, 1963, p. 972-989.

7. Martin, G. T., "An Accelerated Euclidean Algorithm for Integer Linear Programming," Recent Advances in Mathematical Programming (R. L. Graves and P. Wolfe, eds.), 1963.

8. Clark, G. and Wright, J. W., "Scheduling of Vehicles from a Central Depot to a Number of Delivery Points, " Operations Research, Vol. 11, 1963, p. 568-581.

9. Lin, Shen, "Computer Solutions of the Traveling Salesman Problem," Bell Syst. Tech. J., Vol. 44, 1965, p. 2245-2269.

10. Christofides, N. and Eilon, S., "An Algorithm for the Vehicle Dispatching Problem, " Operational Research Quarterly, Vol. 20, 1969, p. 309-318.

11. Dantzig, G. B. and Ramser, J. H., "The Truck Dispatching Problem," Operations Research, Vol. 12, 1959, p. 80-91.

12. Gaskell, T. J., "Bases for Vehicle Fleet Scheduling," Operational Research Quarterly, Vol. 18, 1967, p. 281-295.

13. Tillman, F. A. and Cochran, H., "A Heuristic Approach for Solving the Delivery Problem," Journal of Industrial Engineering, Vol. 19, 1968, p. 354-358.

14. Hayes, Robert, "The Delivery Problem," Carnegie Inst. of Tech., Graduate School of Industrial Administration, Management Sciences Research Report No. 106, 1967.

15. Kolesar, P. J., "A Branch and Bound Algorithm for the Knapsack Problem," Management Science, Vol. 13, 1967, p. 723-735.

16. Lawler, E. L. and Wood, D. E., "Branch and Bound Methods: A Survey," Operations Research, Vol. 14, 1966, p. 669-719.

17. Croes, G. A., "A Method for Solving Traveling Salesman Problems," Operations Research, Vol. 6, 1958, p. 791-812.

18. Karg, Robert L. and Thompson, Gerald L., " A Heuristic Approach to Solving Traveling Salesman Problems," Management Science, Vol. 10, 1963, p. 225-248.

19. Greene,Joseph, "A Coordinate Oriented Algorithm for the Traveling Salesman Problem," Ph.D. Thesis, University of Missouri-Rolla, Rolla, Missouri, 1970.

VITA

Leland Ray Miller was born on May 20, 1938, in Wooster, Ohio. He received his secondary education at Pandora-Gilboa High School in Pandora, Ohio. After attending Ohio Northern University at Ada, Ohio, for one year, he completed three years at Bluffton College in Bluffton, Ohio, whereupon he received his Bachelor of Science degree with a major in mathematics.

From September 1960 to June 1964, he taught mathematics at Fostoria Public High School in Fostoria, Ohio. During the summer of 1961 he was granted a N.S.F. fellowship to attend a Mathematics Institute at Kent State University. In August of 1964, he received his Master of Arts Degree in Mathematics at Bowling Green State University, under a N.S.F. Sequential Institute. The following year he was granted a N.S.F. Academic Institute for Mathematics Supervisors at Bowling Green State University, whereupon he received his Specialist Degree in Mathematics Education.

He taught at Bluffton College from 1965 to 1968 where he was Assistant Professor of Mathematics. During the summers of 1967, 1968, and 1969, he was granted a N.S.F. fellowship to attend a Computer Science Institute at the University of Missouri-Rolla. He has been employed as Instructor of Mathematics at the University of Missouri-Rolla from September 1968 to the present time.

On August 24, 1963 he was married to the former Judith E. Bowers of Beaverdam, Ohio. They have two children, Craig and Kristen.

APPENDIX A

SWEEP ALGORITHM COMPUTER PROGRAM

The computer program uses the following variables.

N	- number of locations including the depot
C	- load capacity for each vehicle
XD	- distance constraint for each vehicle
XLD	- additional distance per location
X(1), Y(1)	- rectangular coordinants for the depot
Q(I)	- demand for location I
X(I), Y(I)	- rectangular coordinants for location I
A(I,J)	- shortest distance from location I to location J

The following data are required for each data set.

First data card;

columns 1 - 5	N	I5
columns 6 - 15	C	F10.2
columns 16- 25	XD	F10.2
columns 26- 35	XLD	F10.2

Data for cards 2 through N+1

columns 1 - 10	X(I)	F10.5
columns 11- 20	Y(I)	F10.5
columns 21- 30	Q(I)	F10.5

```
C  SWEEP ALGORITHM FOR THE VEHICLE DISPATCH PROBLEM
C  FORWARD PROCEDURE
C  A(I,J) IS DISTANCE FROM LOCATION I TO LOCATION J
C  C IS THE LOAD CAPACITY
C  XD IS THE DISTANCE CONSTRAINT
C XLD IS ADDED DISTANCE PER STOP
      COMMON A(101,101), IROUT(101)
      DIMENSION X(101), Y(101), Q(101), R(101), S(101),
     1 SS(101), MK(101), NT(101), KK(101),K(101)
      READ (1,255) N,C,XD,XLD
  255 FORMAT (I5,3F10.2)
      AVQ = 0
      DO 1 I = 1,N
      AVQ = AVQ + Q(I)
    1 READ (1,256) X(I), Y(I), Q(I)
  256 FORMAT (3F10.5)
      AVQ = AVQ/(N-1)
      XX = X(1)
      YY = Y(1)
      KLN = 1
      KV = 0
C  CHANGE TO POLAR COORDINATES WITH DEPOT AT ORIGIN
      WRITE (3,200)
  200 FORMAT ('1',18X,'X(I)', 7X,'Y(I)',5X,'DEMAND',4X,
     1 'RADIUS',4X,'ANGLE')
      RMAX = 0.
      SUMR = 0.
      DO 2 I = 2,N
      R(I) = SQRT((X(I) - XX)**2 + (Y(I) - YY)**2)
      S(I) = ATAN2(Y(I) - YY,X(I) - XX)
      SUMR = SUMR + R(I)
      WRITE (3,257) I,X(I), Y(I), Q(I), R(I), S(I)
  257 FORMAT (8X,I3,5(4X,13F10.3))
      IF(RMAX - (R(I))) 66,2,2
   66 RMAX = R(I)
```

```
    2 CONTINUE
      AVR = SUMR/(N-1)
      DO 81 I = 1,N
      DO 81 J = I,N
      A(I,J) = SQRT ((X(I) - X(J))**2 + (Y(I) - Y(J))**2)
   81 A(J,I) = A(I,J)
      K(1) = 1
      K(N+1) = 1
      MM = 1
C ARRANGE LOCATIONS IN ASCENDING ORDER
   21 J = N
      SUMD = 0.
      DO 67 I = 2,N
      K(I) = I
   67 SS(I) = S(I)
    5 XMAX = -1000000.
      DO 3 I = 2,J
      IF(SS(I) - XMAX)3,3,4
    4 XMAX = SS(I)
      II = I
    3 CONTINUE
      IB = K(II)
      K(II) = K(J)
      K(J) = IB
      B = SS(II)
      SS(II) = SS(J)
      SS(J) = B
      J = J - 1
      IF(J-2) 6,6,5
    6 CONTINUE
C FORMING THE FIRST ROUTE
   11 J = 2
      M = 1
      KCECK = 0
      N1 = 0
```

```
      N2 = 0
      LX = 0
      JJ = 2
      SUM = Q(K(2))
      WRITE (3,201) MM
  201 FORMAT (///30X,'ROUTES NUMBER ',I5)
      WRITE (3, 258) (K(I), I = 2,N)
  258 FORMAT (40I3)
      MM = MM + 1
   12 J = J + 1
   45 IF (SUM + Q(K(J)) - C) 13, 13,14
   13 SUM = SUM + Q(K(J))
      KCECK = 0
  792 IF(J-N) 12,27,27
   14 CONTINUE
  714 JJJ = J - 1
C     CHECKING NEXT LOCATION
C     FINDING TWO NEAREST POINTS
  328 F = 1000000.
      DO 40 I = JJ,JJJ
      EFG = R(K(I)) - S(K(I)) * AVR
      IF (F - EFG) 40,40,48
   48 F = EFG
      KII = I
   40 CONTINUE
      RX = 100000000.
      DO 346 I = 1,4
      JX = J - I
      IF(JX .LT.2) GO TO 346
      IF(R(K(JX))/AVR - .7) 346,346,347
  347 J5 = J + 5
      IF (J5 - N) 363, 363,364
  364 J5 = N
  363 DO 348 II = J,J5
      IF (A(K(JX),K(II)) - RX) 349,348,348
```

```
  349 RX = A(K(JX), K(II))
      JJX = JX
      JII = II
  348 CONTINUE
  346 CONTINUE
C  CHECK THE MODIFIED P-SECT DESPERSEMENT
      IF(KCECK .GT. 0) GO TO 374
      KOUNT = 1
      DO 320 I = JJ,JJJ
      KOUNT = KOUNT + 1
  320 IROUT(KOUNT) = K(I)
      IROUT(1) = 1
      IROUT(KOUNT+1) = 1
      CALL TRAVS (KOUNT,DIST)
      DIST = DIST + (KOUNT - 1) *XLD
      IF (DIST .GT.XD) GO TO 76
      DO 716 I = 1,KOUNT
  716 KK(I) = IROUT(I)
      SUMQ = SUM
  374 CONTINUE
      IF(RX .GT. 100000.) GO TO 75
      RRX = R(K(JII))
      JIX = JII
      DO 334 I = J,JIX
      IF(R(K(I)) - RRX) 334,334,335
  335 RRX = R(K(I))
      JII = I
  334 CONTINUE
   43 IF(SUM + Q(K(JII)) - Q(K(KII)) - C)  44, 44,75
   44 JY = 5
      IF (JY-(N-JJJ))  324,322,322
  322 JY = N - JJJ
  324 JZ = JY + 1
      IF (KCECK .EQ. 1) GO TO 375
      DO 321 I = 2,JZ
```

```
321 IROUT(I) = K(JJJ+I-1)
    IROUT(1) = 1
    CALL BTS (JY, DIST2)
375 CONTINUE
    KCECK = 0
    IF (JII - JJJ + 1 .GT. JY) GO TO 443
    DO 332 I = 2,JZ
332 IROUT(I) = K(JJJ+I-1)
    IROUT(1) = 1
    IROUT (JII-JJJ+1) = K(KII)
    CALL BTS  (JY, DIST3)
    KOUNT = 1
    DO 331 I = JJ,JJJ
    KOUNT = KOUNT + 1
331 IROUT(KOUNT) = K(I)
    IROUT (1) = 1
    IROUT (KOUNT+1) = 1
    IROUT(KII - JJ + 2) = K(JII)
    CALL TRAVS (KOUNT,DIST1)
    DIST1 = DIST 1 + (KOUNT - 1)  * XLD
    IF (DIST1 .GT. XD) GO TO 443
    EFG = AVR  * (Q(K(JII)) - Q(K(KII))) / AVQ
    IF (EFG+DIST + DIST2 - DIST1 - DIST3) 443,443,326
326 DIST = DIST1
    DO 717 I = 1, KOUNT
717 KK(I) = IROUT(I)
    SUMQ = SUM
    JJ1 = JJJ - 1
    SUM = SUM + Q(K(JII)) - Q(K(KII))
    JI = K(KII)
    DO 51 I = KII,JJ1
 51 K(I) = K(I+1)
    IF (JII .NE. JJJ + 1) GO TO 274
    K(JJJ) = K(JJJ + 1)
```

```
      K(JJJ + 1) = JI
      GO TO 275
  274 K(JJJ) = K(JII)
      K(JII) = JI
  275 J = J - 1
      DIST2 = DIST3
      KCECK = 1
      GO TO 12
C   CHECK THE SECOND MODIFIED P-SECT DISPERSEMENT
  443 MAX = 1000000
      IF(J5 - J .LT. 3) GO TO 75
      DO 420 I = J,J5
      IF (I - JII) 421,420,421
  421 IF (MAX - A(K(I),K(JII))) 420,422,422
  422 JKK = I
      MAX = A(K(I),K(JII))
  420 CONTINUE
      IF (SUM + Q(K(JII)) + Q(K(JKK)) - Q(K(KII)) .GT. C)
     1 GO TO 75
      KOUNT = 1
      JZ = 6
      IF(JII - JJJ + 1 .GE. JZ) GO TO 75
      IF(JKK - JJJ + 1 .GE. JZ) GO TO 75
      IF(JZ - (N -JJJ + 1)) 435,436,436
  436 JZ = N - JJJ
  435 DO 431 I = 2,JZ
      IF(I. EQ.JKK - JJJ + L) GO TO 431
      KOUNT = KOUNT + 1
      IROUT(KOUNT) = K(JJJ + I - 1)
  431 CONTINUE
      IROUT(JII - JJJ + 1) = K(KII)
      IROUT(1) = 1
      JT = KOUNT - 1
      CALL BTS (JT,DIST5)
      KOUNT = 1
```

```
      DO 430 I = JJ,JJJ
      KOUNT = KOUNT + 1
      IROUT(KOUNT) = K(I)
  430 CONTINUE
      IROUT(1) = 1
      KOUNT = KOUNT + 1
      IROUT(KOUNT + 1) = 1
      IROUT(KII - JJ + 2) = K(JII)
      IROUT(KOUNT) = K(JKK)
      CALL TRAVS (KOUNT,DIST4)
      DIST4 = DIST4 + (KOUNT - 1) *XLD
      IF(DIST4 .GT. XD) GO TO 75
      IF (DIST + DIST2 - DIST4 - DIST5) 75,433,433
  433 DIST = DIST4
      DO 718 I = 1, KOUNT
  718 KK(I) = IROUT(I)
      SUM = SUM + Q(K(JII)) + Q(K(JKK)) - Q(K(KII))
      SUMQ = SUM
      M5 = JJJ + 4
      JI = K(KII)
      JM = K(J)
      IF(KII .EQ. JJJ) GO TO 794
      JJ1 = JJJ - 1
      DO 434 I = KII,JJ1
  434 K(I) = K(I+1)
      K(JJJ) = K(JII)
      JJJ = JJJ + 1
      K(JJJ) = K(JKK)
      K(JKK) = JI
      IF(JII .EQ. J) GO TO 793
      K(JII) = JI
      K(JKK) = JM
      GO TO 793
  794 K(J) = K(JII)
      K(KII) = K(JKK)
```

```
      JJJ = JJJ + 1
      K(JII) = JM
      K(JKK) = JI
  793 CONTINUE
      KCECK = 2
      GO TO 12
C DELETING ONE FROM ROUTE
   76 JJJ = JJJ - 1
      KOUNT = KOUNT - 1
      J = J - 1
      SUM = SUM - Q(K(J))
      GO TO 328
C ACCEPTING THE ROUTE
   75 SUMD = SUMD + DIST
      KT = JJJ - JJ + 2
      WRITE (3,719) M,SUMQ,DIST,(KK(I),I=1,KT)
  719 FORMAT (/' ROUTE'.I5,' HAS LOAD',F10.2,' WITH
     1 DISTANCE ',F10.2,' IS'/28(1X,I3))
      LX = 0
      M = M + 1
      SUM = Q(K(J))
      JJ = J
   20 IF(KLN-1) 30,31,30
   31 IF(KV-KOUNT) 32,30,30
   32 KV = KOUNT
   30 CONTINUE
      IF(J-N) 12,27,27
   27 KOUNT = 1
      JJJ = J
      IROUT(1) = 1
      DO 82 I = JJ,J
      KOUNT = KOUNT + 1
   82 IROUT(KOUNT) = K(I)
      IROUT(KOUNT + 1) = 1
      CALL TRAVS (KOUNT, DIST)
```

```
      DIST = DIST + (KOUNT - 1) * XLD
      IF(DIST - XD) 83,83,97
   97 J = J + 1
      GO TO 76
   83 CONTINUE
      WRITE (3,719) M,SUM,DIST,(IROUT (I), I = 1 ,KOUNT)
      SUMD = SUMD + DIST
      WRITE (3,84) SUMD
   84 FORMAT(//'TOTAL DISTANCE IS',F15.5)
C  INCREMENT THE ANGLE ONE LOCATION
      KLN = 2
      IF(MM        -KV) 61,50,50
   61 XMIN = 100000000.
      DO 62 I = 2,N
      IF (S(K(I)) - XMIN) 63,62,62
   63 XMIN = S(K(I))
      MI = K(I)
   62 CONTINUE
      S( MI ) = 3.14529 - ABS(S( MI )) + 3.14529
      GO TO 21
   50 CONTINUE
  521 CONTINUE
      STOP
      END

      SUBROUTINE TRAVS (N,DIST)
      COMMON A(101,101), K(101)
      DIMENSION KK(101), KKK(101)
C  3 OPT FOR TRAVELING SALESMAN
      N1 = N + 1
      DO 34 I = 1, N1
   34 KKK(I) = K(I)
   51 IF(N-3) 54,54,53
   53 N1 = N - 1
      N3 = N - 3
```

```
 5 DO 12 KOUNT = 1,N
   DO 32 IK = 1,N3
   K1 = IK + 1
   DO 32 IJ = K1,N1
   D1 = A(K(IK),K(IJ+ 1)) + A(K(1),K(IJ))
   D = A(K(1),K(IJ+1)) + A(K(IK), K(IJ))
   IF (D1 - D) 6,6,7
 6 IA = 8
   D = D1
   GO TO 17
 7 IA = 2
17 IF(D+A(K(IK+1),K(N))-A(K(1),K(N))-A(K(IK),K(IK+1)) -
  1 A(K(IJ),K(IJ+1)) + .001) 9,32,32
32 CONTINUE
   IB = K(N)
   N1 = N - 1
   DO 13 I = 1,N1
13 K(N-I+1) = K(N-I)
   K(1) = IB
12 CONTINUE
   GO TO 2
 9 DO 19 I = 1,N
19 KK(I) = K(I)
   IJ2 = IJ+2
   K1 = IK+1
   K(N) = KK(IJ+1)
   K0 = 0
   IF(IJ2 - N) 36,36,37
36 DO 20 I = IJ2, N
   K0 = K0 + 1
20 K(K0) = KK(I)
37 DO 21 I = K1,IJ
   K0 = K0 + 1
21 K(K0) = KK(I)
   K(N) = KK(IJ+1)
```

```
      IF(IA - 8) 18,15,18
   15 DO 22 I = 1,IK
      KO = KO + 1
   22 K(KO) = KK(I)
      GO TO 14
   18 DO 25 I = 1, IK
      KO = KO + 1
   25 K(KO) = KK(IK+1-I)
   14 CONTINUE
      DO 35 I = 1,N
   35 KKK(I) = K(I)
      GO TO 5
    2 CONTINUE
   54 CONTINUE
      DIST = A(KKK(N),KKK(1))
      DO 30 I = 2,N
   30 DIST = A(KKK(I-1),KKK(I)) + DIST
      RETURN
      END

      SUBROUTINE BTS (N,BOUND)
      COMMON A(101,101), K(101)
      DIMENSION MM(10,10), T(10,10), IT(10), KK(10)
C  BRANCH ALGORITHM FOR DETERMINING MINIMUM DISTANCE OF A
C  ROUTE BEGINNING AT 1 AND ENDING AT K(N)
      DO 21 I = 1,N
      DO 22 J = 1,N
   22 MM(I,J) = 0
   21 IT(I) = 0
      IT(N+1) = N+1
      T(1,1) = 0
      IT(1) = 1
      BOUND = 100000.
      JJ = 1
      I = 1
    1 I = I + 1
```

```
      II = I - 1
      DO 25 L = 1,II
      IF (IT(L)) 25,25,26
   26 MM(I,IT(L)) = 1
   25 CONTINUE
   12 DX = 100000.
      DO 2 J = 2,N
      IF (MM(I,J) .EQ. 1) GO TO 2
      T(I,J) = T(I-1, JJ) + A(K(JJ), K(J))
      IF(T(I,J) .GT. BOUND) GO TO 8
      IF(DX .LT. T(I,J)) GO TO 2
      DX = T(I,J)
      KZ = J
    2 CONTINUE
      IF(DX .GT. 10000.) GO TO 24
   11 IT (I) = KZ
      JJ = KZ
      MM(I,JJ) = 1
      IF(I.LT. N  ) GO TO 1
      GO TO 28
   24 I = I - 1
      IF (I .EQ. 1) GO TO 13
      DX = 100000.
      DO 27 L = 2,N
      If (MM(I,L) .EQ. 1) GO TO 27
      IF (T(I,L) .GT.DX) GO TO 27
      DX = T (I,L)
      JJ = L
   27 CONTINUE
      DO 29 L = 1,N
   29 MM(I+1,L) = 0
      IF (DX .GT. 10000) GO TO 24
      IT (I) = JJ
      MM (I,JJ) = 1
      IF (I. LT. N  ) GO TO 1
```

```
 28 I = I + 1
    T(I,1) = T(I-1,JJ) + A( K(JJ), K(I))
    IF (T(I,1) .GT. BOUND) GO TO 24
    J = 1
    BOUND = T(I,1)
    IF ( N+1 - I) 36,35,36
 35 DO 34 L = 1,I
 34 KK(L) = K(IT(L))
 36 CONTINUE
  8 IT(I) =   J
    GO TO 24
 13 DO 342 I = 1,N
342 K(I) = KK(I)
    RETURN
    END
```

APPENDIX B

EXAMPLE PROBLEMS USING SWEEP ALGORITHM

DETAILS OF PROBLEM 1

NO.	X	Y	Q	NO.	X	Y	Q
2	151	264	1100	3	159	261	700
4	130	254	800	5	128	252	1400
6	163	247	2100	7	146	246	400
8	161	242	800	9	142	239	100
10	163	236	500	11	148	232	600
12	128	231	1200	13	156	217	1300
14	129	214	1300	15	146	208	300
16	164	208	900	17	141	206	2100
18	147	193	1000	19	164	193	900
20	129	189	2500	21	155	185	1800
22	139	182	700				

NUMBER OF LOCATIONS IS 22
DEPOT CO-ORDINATES ARE X = 145 Y = 215
LOAD CAPACITY IS 6000
DISTANCE CAPACITY IS 200
ADDITIONAL DISTANCE PER LOCATION IS 10
THE ROUTES DETERMINED BY THE SWEEP METHOD ARE
ROUTE 1 IS
1 14 20 22 18 15
ROUTE 2 IS
1 16 19 21 17 11 13
ROUTE 3 IS
1 10 8 6 3 2 7 9
ROUTE 4 IS
1 13 11 12 5 4
THE TOTAL DISTANCE IS 584.60

DETAILS OF PROBLEM 2

NO.	X	Y	Q	NO.	X	Y	Q
2	295	272	125	3	301	258	84
4	309	260	60	5	217	274	500
6	218	278	300	7	282	267	175
8	242	249	350	9	230	262	150
10	249	268	1100	11	256	267	4100
12	265	257	225	13	267	242	300
14	259	265	250	15	315	233	500
16	329	252	150	17	318	252	100
18	329	224	250	19	267	213	120
20	275	192	600	21	303	201	500
22	208	217	175	23	326	181	75

NUMBER OF LOCATIONS IS 23

DEPOT CO-ORDINATES ARE X = 266 Y = 235

LOAD CAPACITY IS 4500

DISTANCE CAPACITY IS 240

ADDITIONAL DISTANCE PER LOCATION IS 10

THE ROUTES DETERMINED BY THE SWEEP METHOD ARE

ROUTE 1 IS

1 19 20 22

ROUTE 2 IS

1 21 23 18 15

ROUTE 3 IS

1 13 7 2 3 4 16 17

ROUTE 4 IS

1 11 14

ROUTE 5 IS

1 12 10 6 5 9 8

THE TOTAL DISTANCE IS 956.40

DETAILS OF PROBLEM 3

NO.	X	Y	Q	NO.	X	Y	Q
2	218	382	300	3	218	358	3100
4	201	370	125	5	214	371	100
6	224	370	200	7	210	382	150
8	104	354	150	9	126	338	450
10	119	340	300	11	129	349	100
12	126	345	950	13	125	346	125
14	116	355	150	15	126	355	150
16	125	355	550	17	119	357	150
18	115	341	100	19	153	351	150
20	175	363	400	21	180	360	300
22	159	331	1500	23	188	357	100
24	152	349	300	25	215	389	500
26	212	394	800	27	188	393	300
28	207	406	100	29	184	410	150
30	207	392	1000				

NUMBER OF LOCATIONS IS 30

DEPOT CO-ORDINATES ARE X = 162 Y = 354

LOAD CAPACITY IS 4500

DISTANCE CAPACITY IS 240

ADDITIONAL DISTANCE PER LOCATION IS 10

THE ROUTES DETERMINED BY THE SWEEP METHOD ARE

ROUTE 1 IS

1 19 24 11 12 13 18 10 9 22

ROUTE 2 IS

1 23 3 6 5 2 7 4 21

ROUTE 3 IS

1 27 29 28 26 25 30

ROUTE 4 IS

1 15 16 14 8 17 20

THE TOTAL DISTANCE IS 885.30

DETAILS OF PROBLEM 4

NO.	X	Y	Q	NO.	X	Y	Q
2	298	427	700	3	309	445	400
4	307	464	400	5	336	475	1200
6	320	439	40	7	321	437	80
8	322	437	2000	9	323	433	900
10	324	433	600	11	323	429	750
12	314	435	1500	13	311	442	150
14	304	427	250	15	293	421	1600
16	296	418	450	17	261	384	700
18	297	410	550	19	315	407	650
20	314	406	200	21	321	391	400
22	321	398	300	23	314	394	1300
24	313	378	700	25	304	382	750
26	295	402	1400	27	283	406	4000
28	279	399	600	29	271	401	1000
30	264	414	500	31	277	439	2500
32	290	434	1700	33	319	433	1100

NUMBER OF LOCATIONS IS 33

DEPOT CO-ORDINATES ARE X = 292 Y = 425

LOAD CAPACITY IS 8000

DISTANCE CAPACITY IS 240

ADDITIONAL DISTANCE PER LOCATION IS 10

THE ROUTES DETERMINED BY THE SWEEP METHOD ARE

ROUTE 1 IS

1 30 29 17 28 27

ROUTE 2 IS

1 15 18 26 25 24 23 21 22 20 19

ROUTE 3 IS

1 13 6 7 8 9 10 11 33 12 14 16

ROUTE 4 IS

1 32 31 4 5 3 2

THE TOTAL DISTANCE IS 817.30

DETAILS OF PROBLEM 5

NO.	X	Y	Q	NO.	X	Y	Q
2	37	52	7	3	49	49	30
4	52	64	16	5	20	26	9
6	40	30	21	7	21	47	15
8	17	63	19	9	31	62	23
10	52	33	11	11	51	21	5
12	42	41	19	13	31	32	29
14	5	25	23	15	12	42	21
16	36	16	10	17	52	41	15
18	27	23	3	19	17	33	41
20	13	13	9	21	57	58	28
22	62	42	8	23	42	57	8
24	16	57	16	25	8	52	10
26	7	38	28	27	27	68	7
28	30	48	15	29	43	67	14
30	58	48	6	31	58	27	19
32	37	69	11	33	38	46	12
34	46	10	23	35	61	33	26
36	62	63	17	37	63	69	6
38	32	22	9	39	45	35	15
40	59	15	14	41	5	6	7
42	10	17	27	43	21	10	13
44	5	64	11	45	30	15	16
46	39	10	10	47	32	39	5
48	25	32	25	49	25	55	17
50	48	28	18	51	56	37	10

NUMBER OF LOCATIONS IS 51

DEPOT CO-ORDINATES ARE X = 30 Y = 40

LOAD CAPACITY IS 160

NO DISTANCE CAPACITY

ADDITIONAL DISTANCE PER LOCATION IS 0

DETAILS OF PROBLEM 5 (Continued)

THE ROUTES DETERMINED BY THE SWEEP METHOD ARE

ROUTE 1 IS

1 48 5 18 43 20 41 42 14 19

ROUTE 2 IS

1 47 6 50 11 40 34 46 16 45 38 13

ROUTE 3 IS

1 12 3 30 22 17 51 35 31 10 39

ROUTE 4 IS

1 49 27 32 29 4 37 36 21 23 2 33

ROUTE 5 IS

1 7 15 26 25 44 8 24 49 28

THE TOTAL DISTANCE IS 524.60

DETAILS OF PROBLEM 6

NO.	X	Y	Q	NO.	X	Y	Q
2	22	22	18	3	36	26	26
4	21	45	11	5	45	35	30
6	55	20	21	7	33	34	19
8	50	50	15	9	55	45	16
10	26	59	29	11	40	66	26
12	55	65	37	13	35	51	16
14	62	35	12	15	62	57	31
16	62	24	8	17	21	36	19
18	33	44	20	19	9	56	13
20	62	48	15	21	66	14	22
22	44	13	28	23	26	13	12
24	11	28	6	25	7	43	27
26	17	64	14	27	41	46	18
28	55	34	17	29	35	16	29
30	52	26	13	31	43	26	22
32	31	76	25	33	22	53	28
34	26	29	27	35	50	40	19
36	55	50	10	37	54	10	12
38	60	15	14	39	47	66	24
40	30	60	16	41	30	50	33
42	12	17	15	43	15	14	11
44	16	19	18	45	21	48	17
46	50	30	21	47	51	42	27
48	50	15	19	49	48	21	20
50	12	38	5	51	15	56	22
52	29	39	12	53	54	38	19
54	55	57	22	55	67	41	16
56	10	70	7	57	6	25	26
58	65	27	14	59	40	60	21
60	70	64	24	61	64	4	13
62	36	6	15	63	30	20	18
64	20	30	11	65	15	5	28
66	50	70	9	67	57	72	37

DETAILS OF PROBLEM 6 (Continued)

NO.	X	Y	Q	NO.	X	Y	Q
68	45	42	30	69	38	33	10
70	50	4	8	71	66	8	11
72	59	5	3	73	35	60	1
74	27	24	6	75	40	20	10
76	40	37	20				

NUMBER OF LOCATIONS IS 76

DEPOT CO-ORDINATES ARE X = 40 Y = 40

LOAD CAPACITY IS 140

NO DISTANCE CAPACITY

ADDITIONAL DISTANCE PER LOCATION IS 0

THE ROUTES DETERMINED BY THE SWEEP METHOD ARE

ROUTE 1 IS

1 4 25 19 56 26 51 33 45

ROUTE 2 IS

1 13 73 32 40 10 41 18

ROUTE 3 IS

1 27 59 11 39 66 67

ROUTE 4 IS

1 36 15 60 12 54 8

ROUTE 5 IS

1 35 14 55 20 9 47 68

ROUTE 6 IS

1 5 46 16 21 58 28 53

ROUTE 7 IS

1 49 48 37 70 72 61 71 38 6 30

ROUTE 8 IS

1 76 31 75 22 62 29 69

ROUTE 9 IS

1 74 2 44 43 65 23 63 3

ROUTE 10 IS

1 52 17 50 24 57 42 64 34 7

THE TOTAL DISTANCE IS 865.70

DETAILS OF PROBLEM 7

NO.	X	Y	Q	NO.	X	Y	Q
2	41	49	10	3	35	17	7
4	55	45	13	5	55	20	19
6	15	30	26	7	25	30	3
8	20	50	5	9	10	43	9
10	55	60	16	11	30	60	16
12	20	65	12	13	50	35	19
14	30	25	23	15	15	10	20
16	30	5	8	17	10	20	19
18	5	30	2	19	20	40	12
20	15	60	17	21	45	65	9
22	45	20	11	23	45	10	18
24	55	5	29	25	65	35	3
26	65	20	6	27	45	30	17
28	35	40	16	29	41	37	16
30	64	42	9	31	40	60	21
32	31	52	27	33	35	69	23
34	53	52	11	35	65	55	14
36	63	65	8	37	2	60	5
38	20	20	8	39	5	5	16
40	60	12	31	41	40	25	9
42	42	7	5	43	24	12	5
44	23	3	7	45	11	14	18
46	6	38	16	47	2	48	1
48	8	56	27	49	13	52	36
50	6	68	30	51	47	47	13
52	49	58	10	53	27	43	9
54	37	31	14	55	57	29	18
56	63	23	2	57	53	12	6
58	32	12	7	59	36	26	18
60	21	24	28	61	17	34	3
62	12	24	13	63	24	58	19
64	27	69	10	65	15	77	9
66	62	77	20	67	49	73	25
68	67	5	25	69	56	39	36

DETAILS OF PROBLEM 7 (Continued)

NO.	X	Y	Q	NO.	X	Y	Q
70	37	47	6	71	37	56	5
72	57	68	15	73	47	16	25
74	44	17	9	75	46	13	8
76	49	11	18	77	49	42	13
78	53	43	14	79	61	52	3
80	57	48	23	81	56	37	6
82	55	54	26	83	15	47	16
84	14	37	11	85	11	31	7
86	16	22	41	87	4	18	35
88	28	18	26	89	26	52	9
90	26	35	15	91	31	67	3
92	15	19	1	93	22	22	2
94	18	24	22	95	26	27	27
96	25	24	20	97	22	27	11
98	25	21	12	99	19	21	10
100	20	26	9	101	18	18	17

NUMBER OF LOCATIONS IS 101

DEPOT CO-ORDINATES ARE X = 35 Y = 35

LOAD CAPACITY IS 200

NO DISTANCE CAPACITY

ADDITIONAL DISTANCE PER LOCATION IS 0

THE ROUTES DETERMINED BY THE SWEEP METHOD ARE

ROUTE 1 IS

1 93 38 99 101 15 39 45 92 86
94 60

ROUTE 2 IS

1 14 95 96 98 88 43 44 16 58
42 23 74 3 59

ROUTE 3 IS

1 41 22 73 75 76 57 24 68 40
26 56 5

ROUTE 4 IS

1 54 27 13 55 25 30 81 69 4
78 77 29

DETAILS OF PROBLEM 7 (Continued)

ROUTE 5 IS

1	80	79	35	36	72	66	67	21	52	10	82
34	51										

ROUTE 6 IS

1	28	70	2	71	31	33	91	64	65	12	63
11	89	32									

ROUTE 7 IS

1	53	8	83	49	20	50	37	48	47	9	46
84	19										

ROUTE 8 IS

1	90	61	6	85	18	87	17	62	100	97	7

THE TOTAL DISTANCE IS 854.5

DETAILS OF PROBLEM 8

NO.	X	Y	Q	NO.	X	Y	Q
2	-99	-97	6	3	-59	50	72
4	0	14	93	5	-17	-66	28
6	-69	-19	5	7	31	12	43
8	5	-41	1	9	-12	10	36
10	-64	70	53	11	-12	85	63
12	-18	64	25	13	-77	-16	50
14	-53	88	57	15	83	-24	1
16	24	41	66	17	17	21	37
18	42	96	51	19	-65	0	47
20	-47	-26	88	21	85	36	75
22	-35	-54	48	23	54	-21	40
24	64	-17	8	25	55	89	69
26	17	-25	93	27	-61	66	29
28	-61	26	5	29	17	-72	53
30	79	38	8	31	-62	- 2	24
32	-90	-68	53	33	52	66	13
34	-54	-50	47	35	8	-84	57
36	37	-90	9	37	-83	49	74
38	35	- 1	83	39	7	59	96
40	12	48	42	41	57	95	80
42	92	28	22	43	- 3	97	56
44	- 7	52	43	45	42	-15	12
46	77	-43	73	47	59	-49	32
48	25	91	8	49	69	-19	79
50	-82	-14	79	51	74	-70	4
52	69	59	14	53	29	33	17
54	-97	9	19	55	-58	9	44
56	28	93	5	57	7	73	37
58	-28	73	100	59	-76	55	62
60	41	42	90	61	92	40	57
62	-84	-29	44	63	-12	42	37
64	51	-45	80	65	-37	46	60
66	-97	35	95	67	14	89	56

DETAILS OF PROBLEM 8 (Continued)

NO.	X	Y	Q	NO.	X	Y	Q
68	60	58	56	69	-63	-75	9
70	-18	34	39	71	-46	-82	15
72	-86	-79	4	73	-43	-30	58
74	-44	7	73	75	- 3	-20	5
76	36	41	12	77	-30	-94	3
78	79	-62	8	79	51	70	31
80	-61	-26	48	81	6	94	3
82	-19	-62	52	83	-20	51	99
84	-81	37	29	85	7	31	12
86	52	12	50	87	83	-91	98
88	- 7	-92	4	89	82	-74	56
90	-70	85	24	91	-83	-30	33
92	71	-61	45	93	85	11	98
94	66	-48	4	95	78	-87	36
96	9	-79	72	97	-36	4	26
98	66	39	71	99	92	-17	84
100	-46	-79	21	101	-30	-63	99
102	-42	63	33	103	20	42	84
104	15	98	74	105	1	-17	93
106	64	20	25	107	-96	85	39
108	93	-29	42	109	-40	-84	77
110	86	35	68	111	91	36	50
112	62	- 8	42	113	-24	4	71
114	11	96	85	115	-53	62	78
116	-28	-71	64	117	7	- 4	5
118	95	- 9	93	119	- 3	17	18
120	53	-90	38	121	58	-19	29
122	-83	84	81	123	- 1	49	4
124	- 4	17	23	125	-82	- 3	11
126	-43	47	86	127	6	- 6	2
128	70	99	31	129	68	-29	54
130	-94	-30	87	131	-94	-20	17

DETAILS OF PROBLEM 8 (Continued)

NO.	X	Y	Q	NO.	X	Y	Q
132	-21	77	81	133	64	37	72
134	-70	-19	10	135	88	65	50
136	2	29	25	137	33	57	71
138	-70	6	85	139	-38	-56	51
140	-80	-95	29	141	- 5	-39	55
142	8	-22	45	143	-61	-76	100
144	76	-22	38	145	49	-71	11
146	-30	-68	82	147	1	34	50
148	77	79	39	149	-58	64	6
150	82	-97	87	151	-80	55	83
152	81	-86	22	153	39	-49	24
154	-67	72	69	155	-25	-89	97
156	-44	-95	65	157	32	-68	97
158	-17	49	79	159	93	49	79
160	99	81	46	161	10	-49	52
162	63	-41	39	163	38	39	94
164	-28	39	97	165	- 2	-47	18
166	38	8	3	167	-42	- 6	23
168	-67	88	19	169	19	93	40
170	40	27	49	171	-61	56	96
172	43	33	58	173	-18	-39	15
174	-69	19	21	175	75	-18	56
176	31	85	67	177	25	58	10
178	-16	36	36	179	91	15	84
180	60	-39	59	181	49	-47	85
182	42	33	60	183	16	-81	33
184	-78	53	62	185	53	-80	70
186	-46	-26	79	187	-25	-54	98
188	69	-46	99	189	0	-78	18
190	-84	74	55	191	-16	16	75
192	-63	-14	94	193	51	-77	89
194	-39	61	13	195	5	97	19
196	-55	39	19	197	70	-14	90

DETAILS OF PROBLEM 8 (Continued)

NO.	X	Y	Q	NO.	X	Y	Q
198	0	95	35	199	-45	7	76
200	38	-24	3	201	50	-37	11
202	59	71	98	203	-73	-96	92
204	-29	72	1	205	-47	12	2
206	-88	-61	63	207	-88	36	57
208	-46	- 3	50	209	26	-37	19
210	-39	-67	24	211	92	27	14
212	-80	-31	18	213	93	-50	77
214	-20	- 5	28	215	-22	73	72
216	- 4	- 7	49	217	54	-48	58
218	-70	39	84	219	54	-82	58
220	29	41	41	221	-87	51	98
222	-96	-36	77	223	49	8	57
224	- 5	54	39	225	-26	43	99
226	-11	60	83	227	40	61	54
228	82	35	86	229	-92	12	2
230	-93	-86	14	231	-66	63	42
232	-72	-87	14	233	-57	-84	55
234	23	52	2	235	-56	-62	18
236	-19	59	17	237	63	-14	22
238	-13	38	28	239	-19	87	3
240	44	-84	96	241	98	-17	53
242	-16	62	15	243	3	66	36
244	26	22	98	245	-38	-81	78
246	70	-80	92	247	17	-35	65
248	96	-83	64	249	-77	80	43
250	-14	44	50				

NUMBER OF LOCATIONS IS 250

DEPOT CO-ORDINATES ARE X = 0 Y = 0

LOAD CAPACITY IS 500

DISTANCE CAPACITY IS 310

ADDITIONAL DISTANCE PER LOCATION IS 0

DETAILS OF PROBLEM 8 (Continued)

THE ROUTES DETERMINED BY THE SWEEP METHOD ARE

ROUTE 1 IS

1 97 74 199 19 138 229 54 66 174 55 205

ROUTE 2 IS

1 113 28 218 84 207 221 37 184 196

ROUTE 3 IS

1 59 151 107 190 231 171 3 9

ROUTE 4 IS

1 191 126 149 27 10 154 249 122 90

ROUTE 5 IS

1 164 65 194 102 115 168 14 204 225 70

ROUTE 6 IS

1 178 83 58 215 236 158 250 238

ROUTE 7 IS

1 119 123 44 224 226 11 43 239 12 242 63

ROUTE 8 IS

1 136 39 57 169 104 114 81 195 198 243 147

ROUTE 9 IS

1 4 85 40 67 48 56 18 176 177 234 16

ROUTE 10 IS

1 220 137 227 79 25 41 128 202 33 76

ROUTE 11 IS

1 244 52 160 148 68 60 163 53 17

ROUTE 12 IS

1 133 98 30 61 159 135 172 182

ROUTE 13 IS

1 7 166 86 106 211 42 111 110 21 228 170

ROUTE 14 IS

1 223 93 179 118 112 38

ROUTE 15 IS

1 45 24 49 144 15 108 241 99 175 197 237

ROUTE 16 IS

1 23 121 129 46 213 188 94 162 180 201 200
117

DETAILS OF PROBLEM 8 (Continued)

ROUTE 17 IS

```
    1  127  181   51   95  152  248   89   78   92   47  217
   64
```

ROUTE 18 IS

```
    1   26  209  145  185  150   87  246  153
```

ROUTE 19 IS

```
    1  193  219  120  240   36  157  247  142
```

ROUTE 20 IS

```
    1  105    8  161   29  183   96   35   88  189    5  165
```

ROUTE 21 IS

```
    1   82  155   77  156  116  146  187  173
```

ROUTE 22 IS

```
    1  216   22  139  210  100   71  109  245  101
```

ROUTE 23 IS

```
    1   73   32   72  230    2  140  140  203  232  143   69
  235   34
```

ROUTE 24 IS

```
    1   20   80  206  222  130   62   91  212  134    6
```

ROUTE 25 IS

```
    1  214  167  208   31  125  131   50   13  192  186
```

THE TOTAL DISTANCE IS   5794.10

APPENDIX C

BRANCH AND BOUND ALGORITHM COMPUTER PROGRAM

The computer program uses the following variables.

N - number of locations including the depot

A(I,J) - time to travel from location I to location J

IT(I) - I^{th} location in the route

F(I,S) - cost function for location I at time S

The following data are required for each data set.

First data card

columns 1 - 80	N	free format

Remaining data cards

columns 1 - 80	((A(I,J),I=1,N),J=1,N)	free format

```
C  BRANCH METHOD FOR ARBITRARY COST FUNCTION WITH ONE SERVER
C  A(I,J) IS TIME FROM LOCATION I TO LOCATION J
C  F(I,S) IS THE COST FUNCTION FOR LOCATION I
      DIMENSION A(10,10),MM(30,30),IT(30),T(30,30),D(30,30)
     1, G(30,30), XL(50)
      READ, N
      DO 21 I = 1,N
      DO 22 J = 1,N
   22 MM(I,J) = 0.
   21 IT(I) = 0.
      D(1,1) = 0
      T(1,1) = 0
      IT(1) = 1
      IT(N+1) = 1
      BOUND = 100000000.
      READ,A
C  OBTAIN ESTIMATE FOR THE TOTAL TIME FOR THE OPTIMAL ROUTE
      K = 0
      SUM = 0
      JJ = N - 1
      DO 40 I = 1,JJ
      II = I + 1
      DO 40 J = II,N
      K = K + 1
   40 SUM = SUM + A(I,J)
      TIME = SUM *2./3.*(N+1) / K
      DO 41 I = 2,N
   41 XL(I) = F(I,TIME)
C  CALCULATE TOTAL COST FOR THE I TH LOCATION OF THE ROUTE
      JJ = 1
      I = 1
    1 I = I + 1
      II = I - 1
      DO 25 L = 1,II
      IF (IT(L)) 25,25,26
```

```
   26 MM(I,IT(L)) = 1
   25 CONTINUE
   12 DX = -100000000.
      DO 2 J = 2,N
      IF (MM(I,J) .EQ. 1) GO TO 2
      T(I,J) = T(I-1,JJ) + A(JJ,J)
      E = F(J,T(I,J))
      D(I,J) = D(I -1, JJ) + E + A(JJ,J)
      IF(D(I,J) .GT. BOUND) GO TO 8
      G(I,J) = XL(J) - E
      IF (G(I,J)  .LT. DX) GO TO 2
      DX = G(I,J)
      KK = J
    2 CONTINUE
C  CHECK IF THERE ARE MORE LOCATIONS TO CONSIDER
      IF (DX .LT. -10000000.) GO TO 24
   11 IT(I) = KK
      JJ = KK
      MM(I,JJ) = 1
      IF(I.LT. N  ) GO TO 1
      GO TO 28
C  CONTINUE IN A DIFFERENT BRANCH
   24 I = I - 1
      IF (I .EQ. 1) GO TO 13
      DX = -100000000.
      DO 27 L = 2,N
      IF (MM(I,L) .EQ.1) GO TO 27
      IF (G(I,L) .LT.DX) GO TO 27
      DX = G(I,L)
      JJ = L
   27 CONTINUE
      DO 29 L = 1,N
   29 MM (I + 1,L) = 0
      IF (DX .LT.-10000000.) GO TO 24
      IT(I) = JJ
```

```
      MM(I,JJ) = 1
      IF(I .LT. N  ) GO TO 1
C   CALCULATE NEW BOUND
   28 I = I + 1
      T(I,1) = T(I-1,JJ) + A(JJ,1)
      D(I,1) = D(I-1,JJ) + A(JJ,1)
      IF(D(I,1) .GT. BOUND) GO TO 6
      BOUND = D(I,1)
      GO TO 6
    8 IT(I) = J
    6 WRITE (3,30) (IT(J), J = 1,I)
   30 FORMAT (/1X,11I10)
      WRITE (3,31) (D(J,IT(J)), J = 1,I)
   31 FORMAT (1X,11F10.2)
      GO TO 24
   13 STOP
      END
```