A GRAPH-THEORETIC APPROACH TO DELIVERY PROBLEMS

by

PRAKASH BHATT

B. Tech. (Mech.), Indian Institute of Technology, Bombay, India, 1970

A MASTER'S THESIS

submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

Department of Industrial Engineering

KANSAS STATE UNIVERSITY

Manhattan, Kansas

1972

Approved by:

Major Professor

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

THIS IS AS RECEIVED FROM THE CUSTOMER.

To my parents whose untiring efforts, guidance,

and encouragement helped me to be what I am today.

ILLEGIBLE DOCUMENT

THE FOLLOWING DOCUMENT(S) IS OF POOR LEGIBILITY IN THE ORIGINAL

THIS IS THE BEST COPY AVAILABLE

ACKNOWLEDGEMENT

The most difficult task in such undertaking is to properly recognize the individuals who aided in its successful completion. The debt incurred can never be repaid, only acknowledged.

I am deeply indebted to Dr. Said Ashour, Associate Professor, Industrial Engineering Department for his invaluable help and constant guidance not only in the preparation of this paper, but also in my graduate studies. Association with him has been both a great pleasure and a liberal education.

I must also express my thanks to Dr. L. E. Grosh and other members of the Industrial Engineering faculty for their continual interest in this work as well as my graduate studies in general.

I would like also, to thank Dr. Konz, Dr. Duncan and Prof. Smaltz for their contributions as members of my committee.

A special word of thanks is included for my friends who lent invaluable support, perhaps unknowingly, in some way or the other, throughout my studies.

I want to express thanks to Ms. Marie Jirak for her excellent typing job.

Also, I could not in good conscience, leave this space void of acknowledgement of Pragnya, my friend, because "They also serve who only stand and wait - Milton".

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

In many practical problems, there exists a large number of feasible solutions. Such problems are said to be of a combinatorial nature. Only for a very few problems, there are mathematical treatments to pinpoint the optimal or near-optimal solution. This then leaves no choice except complete enumeration and evaluation of all alternate solutions. Due to the combinatorial nature of alternatives, large size problems become practically impossible to solve. In general, combinatorial problems have the following common characteristics: (1) a large number of alternate feasible solutions; (2) a decision criterion to evaluate all alternate feasible solutions; and (3) the decision made ensures the best of the different alternatives with respect to the decision criterion.

To illustrate, consider the seemingly simple task of routing carriers from a terminal to n demand points. The total number of solutions is more than (n!). If we consider the set of all solutions, we can further decompose such a set into smaller subsets with respect to two considerations. The first consideration involves the constraints on the problem in the form of demands at various demand points, carrier capacities and number of carriers, among others. Consequently, any route from the total number of routes that violates such requirements would be non-feasible and can be eliminated from consideration as a possible solution. The second consideration is that of the quality of solution. Obviously, there are solutions to the problem which are superior to others with respect to the decision criterion.

THIS BOOK CONTAINS NUMEROUS PAGES WITH ILLEGIBLE PAGE NUMBERS THAT ARE CUT OFF, MISSING OR OF POOR QUALITY TEXT.

THIS IS AS RECEIVED FROM THE CUSTOMER.

Many problems such as project scheduling, shop scheduling, traveling salesman, carrier routing, and line balancing, to mention a few, can be classified as combinatorial problems. The aim of this study is to investigate carrier routing problems. The carrier routing problem is one of determining routes so that some measure of performance, say cost or distance of travel, is optimized and the restrictions on the system are satisfied. The truck routing problem, in its simplest form, would be to determine the routes such that distance travelled is a minimum and the following conditions are satisfied: (1) all demands are met; (2) carrier is not loaded with more than its capacity; and (3) the only available number of trucks are used.

In practice, this simple form of problem may be complicated by one or more added characteristics. For instance, there may be dead-line on time for delivery at particular demand points, and/or delivery can not be made before some specified time. Other type of characteristics can be that of occurance of practical difficulties such as those imposed by labor union. The driver should not drive more than certain number of hours everyday. When there are more than one supply points, the form of the problem changes considerably and these new types of problems are referred to as multi-terminal delivery problems. In the above discussion, the demands are assumed to be known. A new category of carrier routing problems evolves when demands at various demand points are subject to certain probability. The solution to carrier routing problem with known demands, carrier capacities and number of carriers is the principle objective of the study.

1.1 Literature survey

The carrier routings problem may be regarded as a generalization of the classical traveling salesman problem. Although literature on traveling salesman problem is in abundance, very little can be found which directly relates to the carrier routing problem. The principle methods used so far are dynamic programming, branch-and-bound, integer programming and heuristic programming.

Dynamic programming approach has been used by Tillman [18] to solve the school bus scheduling problem. This approach gives an optimal solution to very small scale problems. The approach can be considered that of pure search and so computational effect would limit its practicability. Other dynamic programming approaches, found in literature, are the computational algorithms by Gonzales [9] and Held and Karp [10]. Gonzales has investigated his algorithm by solving small scale problems and found that the computational time grows somewhat faster than exponential with a number of demand points. The same problem has been faced by Held and Karp. It is noted that in addition to computation time, storage requirements might become prohibitive in large size problems.

Integer programming approach has been applied to the carrier routing problem. Miller, Tucker, and Zemlin [14] have attempted at solving this problem using an integer programming model. However, due to limitations in computational feasibility of integer programming procedures at the time, the authors have failed to achieve satisfactory results. Hoping that with the development of efficient integer programming algorithms, such an approach would be practical. Balinski and Quandt [1] have

developed an integer programming model for carrier routing problem using "cutting plane" algorithm developed by Gomory [7,8]. But the method becomes undesirable as for large size problems, the number of variables becomes too large. At the present time, integer programming approach seems to be limited to very small size problems.

The carrier routing problem has been approached by heuristic programming,some of which are based on reliable heuristics such as branch-and-bound. In the branch-and-bound approach of Little et al. [13], pairs of demand points are successively synthesized until ultimately a feasible tour is generated. The procedure then systematically backs up and continues with different pairs, again proceeding toward another feasible solution. It should be pointed out that, during the generalization process, no feasible solution resulting from the pairs presently synthesized can possibly be better than the best feasible solution discovered so far. The process immediately backtracks, continuing synthesis with a different pair. The procedure stops when all possible solutions have been considered, at least implicitly.

The concept of the second type of heuristics is based on cost savings. In order to follow the discussion on various heuristic programming approaches which are based on cost savings, it is convenient to introduce the concept of "saving of distance". Consider the feasible allocation of trucks to demand points (k) and (ℓ). Initially demand points (k) and (ℓ) are served by terminal (T) exclusively. That is, demand points (k) and (ℓ) are linked to terminal (T). The demands are met with by allocating two trucks, one to each demand point. The total distance for all routes is:

$$2d(T,k) + 2d(T,\ell).$$

By linking demand points (k) and (ℓ), the distance for feasible routes is:

$$d(T,k) + d(T,\ell) + d(k,\ell)$$

Therefore, the resulting saving in the total distance over the initial allocation is:

$$d(T,k) + d(T,\ell) - d(k,\ell).$$

In all heuristic programming approaches which are based on cost saving, the saving in distance for each pair of demand points is calculated. The various heuristic programming approaches of cost saving differ in their approaches depending on their emphasis on particular type of constraints.

The first heuristic programming formulation by Dantzig and Ramser [6] assigns each demand point to a route of its own (each demand point is exclusively served by a terminal) and then proceeds by synthesizing pairs of routes, stage by stage, onto single routes using the distance matrix to find potential links based upon the distance saved. The first stage assignment is made according to the constraint that the demand of the pair does not exceed $C/2^{N-1}$, where N is the total number of stages that will be required, and C is the carrier capacity. The number of stages is determined as

$$N = \log_2 t$$

where

$$\sum_{k=1}^{t} q_k \leq C,$$

$$\sum_{k=1}^{t+1} q_k \geq C,$$

and where q_k is the demand at an individual demand point. The resulting synthesis of the first stage becomes the input to the second stage and synthesis of the second stage becomes the input to the third stage and so on until the final stage where the resulting synthesis defines the completed routes. At each subsequent stage, a classical traveling salesman problem must be solved in order to minimize the total distance so that at the final stage the minimum route length is obtained.

Following this work, Clarke and Wright [4] have developed an algorithm to yield a near-optimal solution. Problems with different carrier capacities are also considered, since they felt that Dantzig and Ramser have given too much emphasis on vehicle loading and not enough emphasis on minimizing the total distance. Clarke and Wright have removed the $C/2^{N-1}$ constraints applied in the first stage and replaced it with the requirement that the capacity, C is not to be exceeded at any stage. Their modified algorithm assigns each demand point to a route of its own in the first stage. Routes are then combined stage by stage through maximizing the savings achieved by linking two of these routes, subject to the capacity constraints. Thus, the final solution is achieved when no more linking of routes is feasibly possible. This method produces better solutions than that of Dantzig and Ramser. The solution can be improved by solving a classical traveling salesman problem for each route. This algorithm can be used to solve large size problems with relatively less computational effort involved.

Tillman and Cochran [16] have modified Clarke and Wright's method to incorporate additional constraints on the system and to improve the computational procedure. The modified algorithm by Tillman and Cochran is effective in solving problems with restrictions limiting the total miles that can be travelled on a given route. The truck reassignment or re-allocation in their algorithm has been designed to utilize the carriers which have been displaced from the initial allocation by a new combination of demand points, through reassigning the carriers displaced to loads after each pair of demand points have been combined. Each new combined demand is then assigned to the carrier of the smallest capacity which are capable of carrying this demand. This modification helps make better utilization of available carriers. The other modification incorporated is the extended look-ahead feature from the selection of points on a route.

The last heuristic programming procedure which can be found in the literature is an algorithm by Herring [12] which investigates various heuristic look-ahead rules. The proposed algorithm with 3-decision look-ahead method, fails to show consistency in results. It has been suggested that a 6-decision look-ahead method would eliminate the difficulties encountered in the 3-decision look-ahead method; however an excessive computational effort would be required.

1.2 Proposed Research

The motivation for beginning this research was due to the fact that, until present time, graph theory has not been applied to delivery problems.

The apparent scarcity of work dealing with graph theory instigated research into two basic areas. They are (1) the development of a general

graph-theoretic algorithm for both multi-terminal and single-terminal delivery problems, based on out-of-kilter concept, and (2) the investigation into the computational experience to test the feasibility and computational efficiency of the algorithm by solving the problems of varying dimensions.

CHAPTER II

A SINGLE-TERMINAL GRAPH-THEORETIC ALGORITHM

The single-terminal delivery problem in its simplest form, and with which this chapter is concerned, is basically one of determining routes for a fleet of carriers, delivering a homogeneous commodity from a single terminal to a geographical array of demand points such that the total miles travelled is minimized. In solving this problem, the proposed approach requires the following conditions to be satisifed: (1) the distance matrix is symmetrical; (2) the requirement at each demand point is known and must be fulfilled; (3) the various routes are mutually exclusive, that is, the given demand point must not appear on more than one route; (4) the number and capacities of available carriers are known; and (5) the routes to be determined must all be either "pick up" or "delivery" routes and not both. This chapter is devoted to discussing the basic concepts of a graph-theoretic approach, illustrating it with a sample problem, and stating a computational algorithm in formal steps.

2.1 Basic concepts:

The graph-theoretic approach consists of two main phases. Phase I determines the minimum route including the terminal, and phase II decomposes the obtained route into subtours to satisfy the various constraints such as the capacity of carriers and the number of available carriers. In order to follow the discussion of the basic concepts, it is convenient to introduce a number of terms and their definitions.

Spanning tree. A spanning tree is a connected graph on nodes 1,2, ..., n without cycles. A spanning tree associated with the minimum cost is referred to as a minimum spanning tree.

1-tree. A 1-tree consists of a spanning tree on nodes 1,2, ..., k-1, k+1, ..., n together with two arcs incident with node (k). A 1-tree associated with the minimum cost is referred to as a minimum 1-tree.

Degree of a node. A degree of a node is the number of arcs emanating from and to the node.

Out-of-kilter node. A node (k) is said to be out-of-kilter, if for all $\{(k,\ell)\} \in A$, $V_k = (\delta_k - 2)$, $V_k \neq 0$ where $\{(k,\ell)\}$ is a set of arcs of minimum 1-tree, A is a minimum 1-tree, and δ_k is the degree of node (k). Node (k) is said to be out-of-kilter higher, if $V_k \geq 1$ and out-of-kilter low if $V_k = -1$.

Tour. A tour is a 1-tree where each of the nodes is in-kilter (that is, each node has a degree of 2; one directed arc to the node and other from it).

Cut at arc (k,ℓ). A cut at arc (k,ℓ) is breaking of arc (k,ℓ) in a minimum tour and connecting individual nodes (k) and (ℓ) to terminal node (T). The increase in cost (or distance) due to cut at a particular arc (k,ℓ) is:

$$\Delta(k,\ell) = d(T,k) + d(T,\ell) - d(k,\ell)$$

where, in general, d(i,j) is the distance traveled between nodes i and j.

The objective in this approach is to find out a tour or a route from the 1-tree. Every tour is a 1-tree and a 1-tree is a tour if, and only if, each of its nodes is in-kilter. Thus, it is evident that if a minimum

1-tree is a tour, it is the "best" route and it is the optimal solution to the carrier routing problem given that there exists a carrier with capacity satisfying all demand points in one tour. Associated with the process of conversion of the minimum 1-tree to the minimum tour is the concept of π vector and the gap function. For the formulation of LP, consider a vector π in which the elements represent the difference between the current distance element $d(k,\ell)$ and the uptated one $\hat{d}(k,\ell)$. Thus the vector π is such that the right choice of π-vector modifies the distance matrix to convert minimum 1-tree to minimum tour.

Gap function. A gap function, $f(\pi)$ is a non-negative function and is equal to the amount by which the cost (or distance) of a minimum tour $d(k,\ell) + \pi_k + \pi_\ell$ exceeds the cost of a minimum 1-tree. That is, if the 1-trees of I_n are indexed 1,2, ..., i, ... q by definition

$$f(\pi) = \omega + 2 \sum_{k=1}^{n} \pi_k - \min_i [D_i + \sum_{k=1}^{n} \pi_k \delta_{ik}]$$

where

ω = cost of the minimum tour with respect to costs $d(k,\ell)$

δ_{ik} = degree of node (k) in the i-th tree.

D_i = cost of the minimum 1-tree.

At this point, it would be appropriate to establish the distinction between the actual cost of travel $d(k,\ell)$ and the modified cost of travel $d(k,\ell) + \pi_k + \pi_\ell$ between nodes (k) and (ℓ). Transformation from arc cost $d(k,\ell)$ to $d(k,\ell) + \pi_k + \pi_\ell$ affects the identity of the minimum 1-tree but as all tours are changed by the same amount, comparison between tours is unaffected. Our chief interest is the gap function $f(\pi)$ as it is the amount

by which the cost of the minimum tour exceeds the cost of the minimum 1-tree. The problem can formally be stated as

Minimize

$$f(\pi) = \omega + 2 \sum_{k=1}^{n} \pi_k - \min_i \{D_i + \sum_{k=1}^{n} \pi_k \cdot \delta_{ik})$$

$$= \omega - \min_i [D_i + \sum_{k=1}^{n} \pi_k [\delta_{ik} - 2]]$$

And it is obvious that minimizing $f(\pi)$ is equivalent to maximizing $f'(\pi)$ where,

$$f'(\pi) = \min_i [D_i + \sum_{k=1}^{n} \pi_k(\delta_{ik}-2)]$$

Dualizing the above problem and putting it in linear programming form, we have

Maximize ω

subject to

and $\omega \leq D_i + \sum_{k=1}^{k=n} \pi_k \cdot V_{ik}$.

$\omega \geq 0$

To solve the above program, branch-and-bound method is used. Central to this approach is the concept of an out-of-kilter node. At each level L, out-of-kilter nodes are identified and the gap between the minimum 1-tree and the minimum tour is reduced by the amount

$$\delta^L(k) = \min [\bar{d}(k,\ell) - d(r,s)]$$

where for out-of-kilter low node (k), arc (k,ℓ) is a substitute for arc (r,s)

which is already present in the minimum 1-tree. Similarly, for out-of-kilter high node (k), the amount by which the gap between the minimum 1-tree and the minimum tour reduces at each level is

$$\delta^{L}(k) = \min [\bar{d}(i,j) - d(r,k)]$$

where arc (i,j) is a substitute for (r,k) which is already present in the minimum 1-tree. The cost (or distance) matrix is updated by the miniumum $\delta^{L}(k)$ at each level L. That is, for the subsequent iterations, the modified distances are used. The distances are modified as

$$\hat{d}(k,\ell) = d(k,\ell) + \delta^{L}(k)$$

where $\hat{d}(k,\ell)$ and $d(k,\ell)$ are the modified and original cost of travel from (k) to (ℓ), respectively. The increment $\delta^{L}(k)$ is either positive or negative depending on whether node (k) is out-of-kilter high or out-of-kilter low.

At any level, if there are arcs which are mutually exclusive for membership of the minimum 1-tree but have the same costs of travel, there exist more than one minimum 1-tree differing from each other by the corresponding candidate arcs. Branching is carried out at a particular arc if at particular level there is a number of minimum 1-trees and no node is "consistently out-of-kilter". An out-of-kilter node is said to be consistently out-of-kilter if it has the same degrees in all the minimum 1-trees at certain level. The bound on the cost of the minimum tour is found out as

$$C^{L} = C^{L-1} + \delta^{L-1}(k)$$

where

C^L = lower bound on cost of the minimum tour at level L

$\delta^{L-1}(k)$ = cost increment resulting due to out-of-kilter node (k)

The initial lower bound at level 0 is

$$C^0 = \sum_{(k,\ell)} d(k,\ell),\ (k,\ell) \in A^L$$

The lower bound associated with the final minimum 1-tree, which represents a tour, becomes the minimum distance of travel.

Having obtained the minimum tour, the constraints of the carrier capacity are applied. If a carrier capacity permits the visit to all demand points in one route, the minimum tour obtained is the solution and no further consideration is necessary. Otherwise, such a tour must be decomposed into feasible subtours to satisfy the constraints of the capacity and number of available carriers. Increment in the distance of the tour is found out for cut at each arc in the minimum tour. This can be formally stated as

$$\Delta(k,\ell) = d(T,k) + d(T,\ell) - d(k,\ell)$$

where $\Delta(k,\ell)$ is the increase in minimum route due to cut at (k,ℓ). The arc chosen for the cut depends on two considerations. The first consideration being whether the cut at an arc with minimum increment produces feasible routes. The second consideration investigates whether the cut distributes the demand points evenly in the sublinks. If any cut with minimum increase satisfies either of the above two conditions it is selected. Otherwise, the cut with the next minimum increase is chosen for consideration.

The procedure is repeated with sublinks till the feasible subtours are formed.

2.2 Sample Problem

The above approach is illustrated in this section by a single-terminal delivery problem with six demand points. The distance matrix, demands at demand points, and number of available carriers along with their capacities are displayed in Table 2.1. The procedure is discussed step by step as follows.

Step 1. Construct the initial minimum 1-tree. First a minimum spanning tree is constructed over the demand points (or nodes) and two minimum arcs are drawn from the terminal to the minimum spanning tree. The minimum spanning tree is constructed by taking the minimum distance arcs from the distance matrix. It is important that there is no loop formed among the subset or complete set of nodes. In construction of the minimum 1-tree, in certain instances, two or more mutually exclusive arcs with the same travel costs, may be candidates for inclusion in the minimum 1-tree. In such instances inclusion of each of such mutually exclusive arcs in the minimum 1-tree will result in two or more distinct minimum 1-trees. These minimum 1-trees will differ only in this one respect. For our sample problem, the associated minimum 1-tree is shown in Figure 2.1. The minimum spanning tree is constructed over the six nodes and is shown distinctly by light lines. By drawing two minimum arcs (thick lines) from node (T) or terminal, the graph becomes a minimum 1-tree. The algorithm is initiated by setting level L = 0. The lower bound on the distance of tour is computed at level 0 such that

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

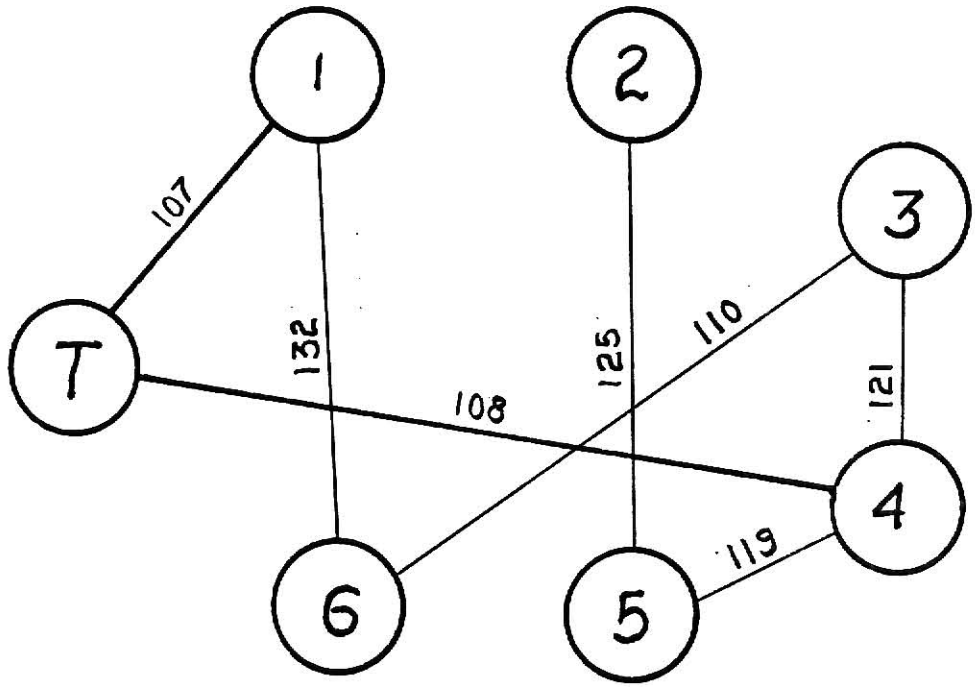
THIS IS AS RECEIVED FROM CUSTOMER.

Table 2.1. The Distance Matrix and Demands for Demand Points, Number and Capacities of the Available Carriers for the Sample Problem.

2	331					
3	152	284				
4	172	167	121			
5	217	125	216	114		
6	132	388	110	227	301	
7	107	224	154	108	113	207
Nodes ↓→	1	2	3	4	5	6
Demands	12	8	15	6	14	8

Table showing available number of carriers with the corr-esponding capacities

Truck	40	30	20
Available number	1	1	2



*Arcs are labelled with associated distances

Figure 2.1. Initial Minimum 1-Tree.

$$C^0 = \sum_{(k,\ell)} d(k,\ell), \qquad (k,\ell) \in A^0$$

where A^0 is the initial minimum 1-tree at level 0. Hence,

$$C^0 = 107 + 108 + 110 + 125 + 121 + 114 + 132$$
$$= 817$$

Step 2. Identify out-of-kilter nodes. If there is only one minimum 1-tree, identify all out-of-kilter nodes. But if there are more than one minimum 1-tree and there is at least one node (k) which is consistently out-of-kilter, then the particular node is chosen for computing cost increment. The third possibility at this stage could be of the number of minimum 1-trees and there is no consistently out-of-kilter node. Here for each minimum 1-tree out-of-kilter nodes are identified and corresponding cost increments are computed in step 3. Figure 2.1 depicting the initial minimum 1-tree shows that node (4) is out-of-kilter high and node (2) is out-of-kilter low.

Step 3. Compute cost increments. If node (k) is out-of-kilter low, the increase in the cost of the minimum 1-tree is computed such that

$$\delta^L(k) = \min_{(k,\ell) \notin A^L} [d(k,\ell) - \max_{(r,s) \in A^L} [d(r,s)]], \quad (k) \neq (r), (s)$$

where $\{(r,s)\}$ is a set of arcs which can possibly be replaced by arc (k,ℓ) without forming a loop among nodes 1,2, ..., n but forms a loop with node (T). In our sample problem, node (2) is out-of-kilter low. The increase in the distance of the minimum 1-tree is computed such that

$$\delta^0(2) = \min_{(2,\ell)\notin A^0} [d(2,\ell) - \max_{(r,s)\in A^0} [d(r,s)]]$$

$$= \min \begin{Bmatrix} d(1,2) - \max \begin{pmatrix} d(1,6) \\ d(3,6) \\ d(3,4) \\ d(4,5) \end{pmatrix} \\ d(2,3) - \max \begin{pmatrix} d(3,4) \\ d(4,5) \end{pmatrix} \\ d(2,4) - \max \begin{pmatrix} d(4,5) \end{pmatrix} \\ d(2,6) - \max \begin{pmatrix} d(3,6) \\ d(3,4) \\ d(4,5) \end{pmatrix} \\ d(2,T) - \max \begin{pmatrix} d(1,T) \\ d(4,T) \end{pmatrix} \end{Bmatrix}$$

$$= \min \begin{Bmatrix} 331 - \max \begin{pmatrix} 132 \\ 110 \\ 121 \\ 114 \end{pmatrix} \\ 284 - \max \begin{pmatrix} 121 \\ 114 \end{pmatrix} \\ 167 - 114 \\ 388 - \max \begin{pmatrix} 110 \\ 121 \\ 114 \end{pmatrix} \\ 224 = \max \begin{pmatrix} 107 \\ 108 \end{pmatrix} \end{Bmatrix} = \min \begin{Bmatrix} 331 - 132 \\ 284 - 121 \\ 167 - 114 \\ 388 - 121 \\ 224 - 108 \end{Bmatrix} = \min \begin{Bmatrix} 199 \\ 163 \\ 53 \\ 267 \\ 116 \end{Bmatrix} = 53.$$

However, if node (k) is out-of-kilter high the increase in the cost of the minimum 1-tree is computed such that

$$\delta^L(k) = \min_{(k,\ell)\in A^L} [\min_{(r,s)\notin A^L} [d(r,s)] - d(k,\ell)], \quad (k) \neq (r), (s)$$

where {(r,s)} is a set of arcs which can possibly replace arc (k,ℓ) without forming a loop among nodes 1,2, ..., n but forms a loop with node (T). Node (4) is out-of-kilter high at level 0 and the increase in the cost of the minimum 1-tree is computed such that

$$\delta^0(4) = \min_{(4,\ell)\in A^0} \left[\min_{(r,s)\notin A^0} [d(r,s)] - d(4,\ell) \right]$$

$$= \min \left\{ \begin{array}{l} \min \begin{pmatrix} d(1,2) \\ d(1,5) \\ d(2,3) \\ d(2,6) \\ d(3,5) \\ d(5,6) \end{pmatrix} - d(3,4) \\ \min \begin{pmatrix} d(1,2) \\ d(1,5) \\ d(2,3) \\ d(2,6) \\ d(3,5) \\ d(5,6) \end{pmatrix} - d(4,5) \\ \min \begin{pmatrix} d(2,T) \\ d(3,T) \\ d(5,T) \\ d(6,T) \end{pmatrix} - d(4,T) \end{array} \right\}$$

$$= \min \left\{ \begin{array}{l} \min \begin{pmatrix} 331 \\ 217 \\ 284 \\ 388 \\ 216 \\ 301 \end{pmatrix} - 121 \\ \min \begin{pmatrix} 331 \\ 217 \\ 284 \\ 388 \\ 216 \\ 301 \end{pmatrix} - 114 \\ \min \begin{pmatrix} 224 \\ 154 \\ 113 \\ 207 \end{pmatrix} - 108 \end{array} \right\} = \min \begin{pmatrix} 216 - 121 \\ 216 - 114 \\ 113 - 108 \end{pmatrix} = \min \begin{pmatrix} 95 \\ 102 \\ 5 \end{pmatrix} = 5$$

Step 4. Select a node having the minimum cost increment. Nodes (2) and (4) are out-of-kilter, and the increments due to each is computed in step 3. Node (4) is selected as the node which yields the minimum increment.

$$\delta^*(k) = \min_{(k)} [\delta^L(k)], \qquad k = 2,4,$$

or

$$\delta^*(4) = 5.$$

If a tie exits for the minimum $\delta^*(k)$, a node is selected by any particular rule. After finding $\delta^*(k)$, level L is increased by one and the lower bound on the cost of tour is updated as

$$C^L = C^{L-1} + \delta^*(k)$$

or

$$C^1 = C^0 + \delta^*(k) \qquad = 817 + 5 = 822.$$

Step 5. Update the distance matrix and associated minimum 1-tree(s). If the selected node (k) is out-of-kilter low, the distance matrix is updated such that

$$\hat{d}(k,\ell) = d(k,\ell) - \delta^*(k), \ (\ell) = 1,2, \ldots, n, T; \ (\ell) \neq (k),$$

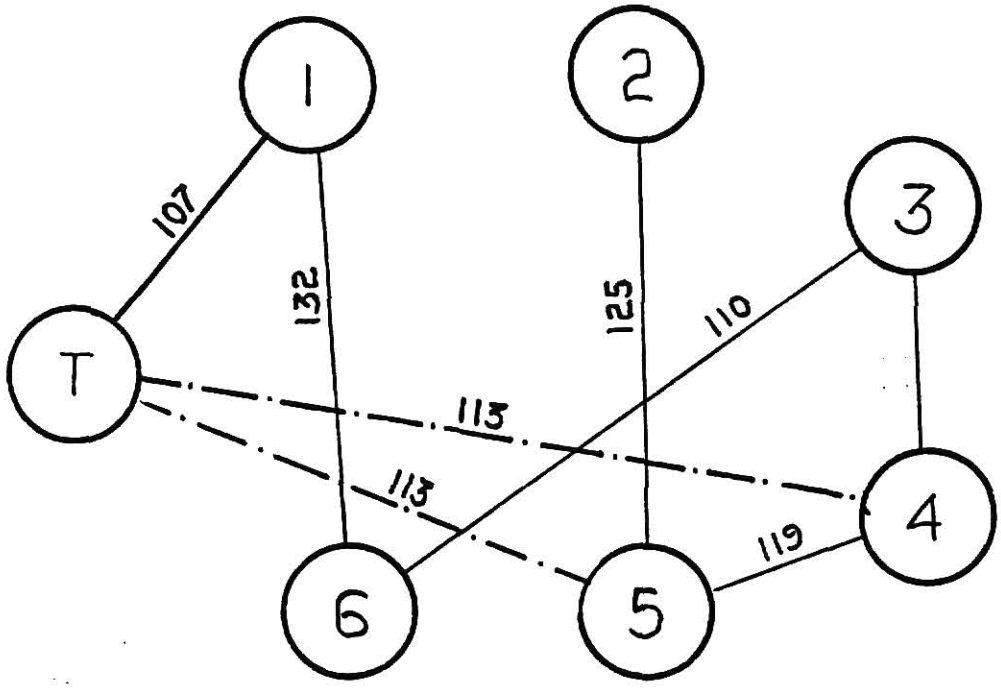
and consequently, a minimum 1-tree(s) is constructed from the updated distance matrix. However, if the selected node (k) is out-of-kilter high, the distance matrix is updated such that

$$\hat{d}(k,\ell) = d(k,\ell) + \delta^*(k), \ (\ell) = 1,2, \ldots, n, T; \ (\ell) \neq (k),$$

and a minimum 1-tree(s) is constructed from the updated distance matrix. As shown earlier, node (4) is out-of-kilter high with an increment of $\delta^*(4) = 5$. The updated distance matrix and corresponding two minimum 1-trees are shown in Table 2.2 and Figure 2.2, respectively. At level 1 two trees

Table 2.2 Updated Distance Matrix at Level L = 1 for the Sample Problem.

2	331					
3	152	284				
4	177	172	126			
5	217	125	216	119		
6	132	388	110	232	301	
T	107	224	154	113	113	207
Nodes	1	2	3	4	5	6



*Hedged lines are the candidate arcs.

Figure 2.2 Minimum 1-Trees at Level L = 1.

are formed as there exists a tie between (4,T) and (5,T) and both the arcs are mutually exclusive as far as their inclusion in minimum 1-tree is concerned. If both are present simultaneously, there are three arcs from node (T) which is against the definition of minimum 1-tree. The two minimum 1-trees differ from each other depending on the candidate arcs (4,T) and (5,T).

Step 6. Check for the minimum tour. If at the particular level all nodes in any minimum 1-tree are in-kilter, the tour is identified and corresponding cost is $C^* = C^L$. However, if no minimum 1-tree is a tour steps 2 through 6 are repeated. At level 1, the minimum 1-trees do not form tours yet and steps 2 through 6 are repeated till any minimum 1-tree is a tour. At level 2, one of the four minimum 1-trees forms a tour such that

T - 5 - 2 - 4 - 3 - 6 - 1 - T

with the total distance of 875. The summary of results for the sample problem is shown in Table 2.4.

Step 7. Decompose the minimum tour into subtours. The increments in the distance of the minimum tour, resulting due to the cut at a particular arc of the minimum tour is computed such that

$$\Delta(k,\ell) = d(k,T) + d(\ell,T) - d(k,\ell)$$

At this point a counter is initiated as K= 0. The increments due to cuts at all arcs are shown in Table 2.5.

Step 8. Check for feasible subtours. The arc with the minimum value of increment at which the cut is made is selected according to the following decision rules: (1) feasible subtours are formed; (2) demand points are evenly distributed in each subtour. If neither of the rules is satisfied,

Table 2.3 Updated Distance Matrix at Level L = 2 for the Sample Problem.

2	278					
3	152	231				
4	177	119	126			
5	217	72	216	119		
6	132	335	110	232	301	
T	107	171	154	113	113	207
Nodes	1	2	3	4	5	6

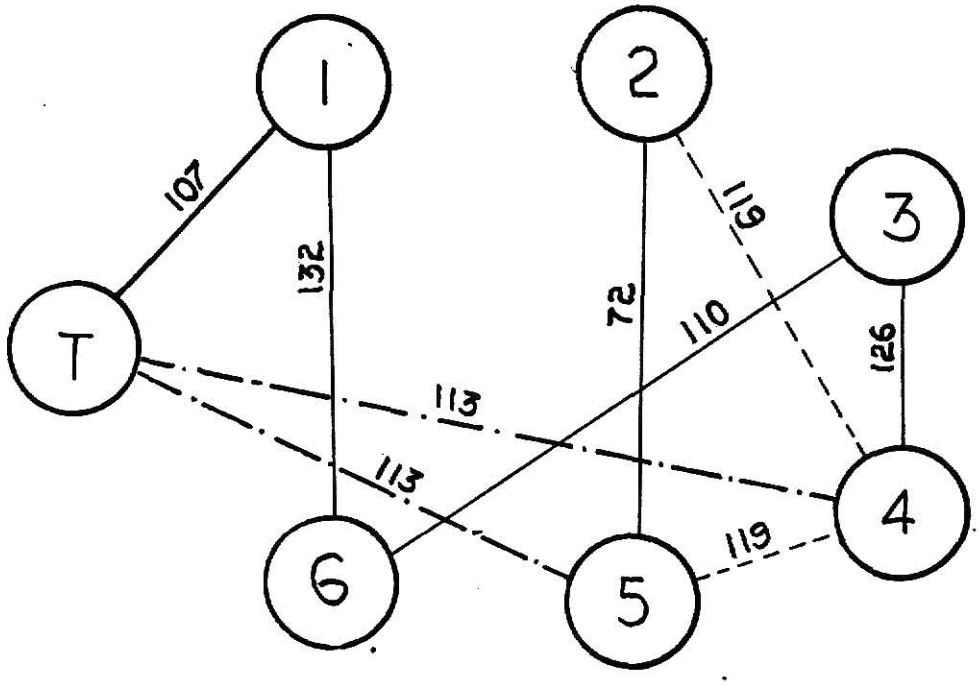


Figure 2.3 Minimum 1-Tree(s) at Level L=2.

Table 2.4 Summary of Results for the Minimum Tour of the Sample Problem.

Level L	Number of Minimum 1-Trees	Candidate Arcs	Arcs Comprising the Minimum 1-Tree $(k,\ell) \in A^L$	Out-of-kilter Nodes High (k)	Out-of-kilter Nodes High $\delta^2(k)$	Out-of-kilter Nodes Low (k)	Out-of-kilter Nodes Low $\delta^L(k)$	Minimum Cost Increment $\delta^*(k)$	Selected Node (k^*)	Lower Bound in the Distance of Minimum Tour C^L
0	1		(T,1),(T,4),(1,6),(2,5) (3,4),(3,6),(4,5)	4	5	2	53	5	4	817
1	2	(4,T)	(T,1),(T,4),(1,6),(2,5) (3,4),(3,6),(4,5)			2	53	53	2	822
		(5,T)	(T,1),(T,5),(1,6),(2,5) (3,4),(3,6),(4,5)							
2	4	(4,T) (4,5)	(T,1),(T,4),(1,6),(2,5) (3,4),(3,6),(4,5)							875
		(4,T) (2,4)	(T,1),(T,4),(1,6),(2,5) (3,4),(3,6),(2,4)							
		(5,T) (4,5)	(T,1),(T,5),(1,6),(2,5) (3,4),(3,6),(4,5)							
		(5,T) (2,4)	T-5-2-4-3-6-1-T**							

** Minimum 1-tree is a tour.

the cut with the next minimum increment is selected and checked for the above two conditions. The procedure is repeated till a cut is found. The cut at arc (3,4) forms two feasible subtours T - 1 - 6 - 3 - T and T - 4 - 2 - 5 - T. The corresponding lower bound on the cost of all subtours is computed as

$$C^* = C^* + \Delta(k,\ell)$$
$$= 875 + 141$$
$$= 1016$$

Step 9. Identify the subtours. If no feasible subtour is formed by certain cut go to step 8 for the non-feasible subtours formed by cut and each non-feasible subtour is treated as a tour. If a feasible subtour (or subtours) is formed, eliminate the nodes from further consideration and counter K is updated by the number of nodes in feasible subtour(s) (or route(s)). If there is any non-feasible subtour, go to step 8. In our sample problem, the first cut at (3,4) results in two feasible subtours T - 1 - 6 - 3 - T and T - 4 - 2 - 5 - T. For this cut counter K is updated such that

$$K = K + N[\Delta(k,\ 0)]$$
$$= 0 + 3 + 3$$
$$= 6$$

where $N[\Delta(k,\ell)]$ is the number of nodes forming feasible subtour(s) when cut at (k,ℓ) is considered.

Step 10. Check for all subtours. If counter K = n, where n is the total number of demand points, the solution to the delivery problem is obtained and all feasible subtours are identified. Otherwise, steps 8 through

Table 2.5 Possible Increments Resulting Due to Cut at a Particular Arc.

No.	Arc for the Cut	Increase Due to Cut $\Delta(k,\ell)$	Bound on all Sub-tours ω^*
1	(1,6)	182	
2	(6,3)	251	
3	(3,4)	141*	875
4	(4,2)	275	
5	(2,5)	212	

*arc chosen for cut

Table 2.6 Capacity and Availability of Trucks at End of First (and Last) Cut.

Truck	40	30	20
Available	1	1	2
Allocated	1	1	0

10 are repeated for all non-feasible subtours. In our sample problem, the first cut results in two feasible subtours and as counter K = 6 the solution to the given delivery problem is obtained. The final solution to the sample problem is shown in Table 2.7 with individual subtour loads and distances and with the total distance of 1016. Table 2.6 shows the availability and allocation of trucks.

2.3 Computational Algorithm

The following is a step by step computational algorithm for solving the single-terminal carrier-routing problem.

Step 1: Construct the initial minimum 1-tree

1.1. Set level L = 0.

1.2. Draw a minimum spanning tree(s) from which a minimum 1-tree(s) is constructed with a cycle around node (T).

1.3. Compute the corresponding lower bound on tour cost such that

$$C^L = \sum_{(k,\ell)} d(k,\ell), \ (k,\ell) \in A^L$$

Step 2: Identify out-of-kilter nodes

2.1. If there is only one minimum 1-tree, identify all out-kilter nodes, and go to step 3.

2.2. If there are more than one minimum 1-tree and there is at least one node (k) which is consistently out-of-kilter, go to step 3 to compute the cost increment for that node only.

2.3. If there are more than one minimum 1-tree and there is no consistently out-of-kilter node, identify out-of-kilter nodes for each minimum 1-tree and go to step 3 to compute the corresponding cost increments.

Table 2.7 Table Showing Final Solution to the Sample Delivery Problem.

No.	Route	Load	Distance	Value of Bound on all Subtours*
1	T - 1 - 6 - 3 - T	35	503	1016
2	T - 4 - 2 - 5 - T	28	513	

*Total distance for all subtours

Step 3: Compute cost increments

3.1. If node (k) is out-of-kilter low, compute the increase in the cost of minimum 1-tree such that

$$\delta^L(k) = \min_{(k,\ell)\notin A^L} [d(k,\ell) - \max_{(r,s)\in A^L} [d(r,s)]], \quad (k) \neq (r),(s)$$

where $\{(r,s)\}$ is a set of arcs which can possibly be replaced by arc (k,ℓ) without forming a loop among nodes 1,2, ..., n but forms a loop with node (T).

3.2. If node (k) is out-of-kilter high compute the increase in the cost of minimum 1-tree such that

$$\delta^L(k) = \min_{(k,\ell)\in A^L} [\min_{(r,s)\notin A^L} [d(r,s)] - d(k,\ell)], \quad (k) \neq (r),(s)$$

where $\{(r,s)\}$ is a set of arcs which can possibly replace arc (k,ℓ) without forming a loop among nodes 1,2, ..., n, but forms a loop with node (T).

Step 4: Select a node having the minimum cost increment

4.1 Find the minimum increment such that

$$\delta^*(k) = \min_{(k)} [\delta^L(k)].$$

If a tie exists select a node by any particular rule.

4.2 Let level L = L+1 and update the lower bound on the cost of tour such that

$$C^L = C^{L-1} + \delta^*(k).$$

Step 5: Update the distance matrix and associated minimum 1-tree(s)

5.1 If the selected node (k) is out-of-kilter low, update the distance matrix such that

$$\hat{d}(k,\ell) = d(k,\ell) - \delta^*(k),\ (\ell) = 1,2,\ \ldots,\ n,\ T;\ (\ell) \neq (k),$$

and construct the corresponding minimum 1-tree(s). Then, go to step 6.

5.2 If the selected node (k) is out-of-kilter high, update the distance matrix such that

$$\hat{d}(k,\ell) = d(k,\ell) + \delta^*(k),\ (\ell) = 1,2,\ \ldots,\ n,\ T;\ (\ell) \neq (k)$$

and construct the corresponding minimum 1-tree(s)

Step 6: Check for the minimum tour

6.1 If all nodes in any minimum 1-tree are in-kilter, identify the tour and the corresponding cost such that

$$C^* = C^L,$$

and go to step 7.

6.2 If no minimum 1-tree consists of all in-kilter nodes, go to step 2.

Step 7: Decompose the minimum tour into subtours

7.1 Compute the increments in the distance of minimum tour due to cut at particular arc such that

$$\Delta(k,\ell) = d(k,T) + d(\ell,T) - d(k,\ell)$$

7.2 Initiate a counter K = 0.

Step 8: Check for feasible subtours

8.1 Select an arc with the minimum value of increment at which the cut is made according to the following conditions: (1) feasible subtours are formed, and (2) demand points are evenly distributed in each subtour. If neither of the conditions is satisfied, the cut with the

next minimum increment is selected and checked for the above two conditions. The procedure is repeated till a cut is found.

8.2 Compute the lower bound on the cost of all subtours such that

$$C^* = C^* + \Delta(k,\ell)$$

Step 9. Identify the subtours

9.1 If any feasible subtour is formed, identify the subtour and eliminate the nodes of the feasible tour from consideration.

9.2 Update the counter such that

$$K = K + N[\Delta(k,\ell)]$$

where $N\ \Delta(k,\ell)]$ is the number of nodes forming a feasible tour when a cut at (k,ℓ).

Step 10. Check for all subtours

10.1 If counter $K = n$, identify all feasible subtours.

10.2 If counter $K < n$, treat the non-feasible subtour as a tour and consider it for cuts by going to step 8.

CHAPTER III

A MULTI-TERMINAL GRAPH-THEORETIC ALGORITHM

The multiple terminal delivery problem can be regarded as a generalization of the single-terminal problem. The major difference as the name suggests is that in the multi-terminal case, there are more than one terminal from which delivery vehicles can be dispatched to carry the commodity to the given demand points. The multi-terminal delivery problem is one of determining the "best" delivery routes with respect to some decision criterion such as cost or distance of travel from the appropriate terminals subject to following conditions: (1) the distance matrix must be symmetrical; (2) all the demands are known and must be met; (3) a given demand point must not appear on more than one route; (4) the number and capacities of available carriers are known; and (5) the routes to be determined must all be either "pick up" or "delivery routes" and not both.

This chapter is devoted to discussing the basic concepts of the proposed approach to solve the multi-terminal delivery problem, illustrating the approach by a sample problem, and stating a computational algorithm in formal steps.

3.1 Basic concepts:

The graph-theoretic approach consists of three main phases. Phase I assigns the various demand points to terminals, the distance being the decision criterion. Phase II finds the minimum tours for each group of demand points including the associated terminal. Phase III decomposes the obtained minimum tours into subtours to satisfy the various constraints

such as the capacity and number of carriers. In other words, the approach differs from the single-terminal delivery problem approach in one respect and that being the association or identification of the demand points with a particular terminal of various terminals.

Central to the approach of associating the demand points with a particular terminal, is the concept of the minimum p-tree. A p-tree is a graph over nodes 1,2, ..., n and i_1, i_2, ..., i_p such that any node (k) is associated with only one of i_r's and there is no loop among any set of nodes 1,2, ..., n. Nodes i_r are termed as origins. A minimum p-tree is a p-tree with minimum cost. In case of multi-terminal delivery problems, the terminals are origins. But here it is not necessary for p to be exactly the same as the total number of terminals. In other words, p can take any integer from 1 to the total number of terminals. After assigning the demand points to a particular terminal, each group of demand points associated with each terminal is treated as single-terminal delivery problem with common carrier capacity and available number of carriers, discussed in the preceding chapter.

3.2 Sample Problem.

The above approach is illustrated in this section by a three terminals and 10 demand points sample problem. The distance matrix, demands at demand points and number of available carriers along with their capacities are displayed in Table 3.1 and Table 3.2. The procedure is discussed step by step as follows:

Step 1. Construct the minimum p-tree. The minimum distance arcs are chosen in sequence subject to the condition that the arc under consideration should be such that the nodes of arc belong to only one of the T_i

Table 3.1 Distance Matrix and Demand for Three Terminals and 10 Demand Points Delivery Problem

2	695									
3	331	502								
4	152	478	284							
5	172	483	167	121						
6	539	272	301	476	367					
7	217	569	125	216	114	394				
8	376	252	258	251	213	250	323			
9	364	290	211	276	192	200	279	70		
10	132	572	388	110	227	608	317	368	408	
Terminal 1 (T_1)	315	388	296	158	186	391	301	151	193	239
Terminal 2 (T_2)	409	336	172	354	234	130	264	170	95	509
Terminal 3 (T_3)	107	571	224	154	108	468	113	321	300	207
Nodes	1	2	3	4	5	6	7	8	9	10
Demands	12	5	8	15	6	7	14	4	13	8

Table 3.2 Table Showing Available Number of Trucks and Corresponding Capacities.

Truck	40	30	20
Available	1	3	10

nodes. From the distance matrix, the first arc chosen is (8,9) with the minimum distance of 70. The second arc chosen is (T_2, 9). Now, nodes (8) and (9) are identified with origin (T_2) and so any node, if considered with this group, can not be associated with any other terminals (other than (T_2)). The third and fourth arcs selected are (T_3, 1) and (T_3, 5). Applying the same argument, any node, if considered with elements of this second group of (1,5), can not be associated with any terminal other than (T_3). The minimum p-tree (2-tree) of the sample problem is shown in Figure 3.1 and corresponding groups of demand points can be recognized as:

Group 1. (T_2), (2), (6), (8), (9).

Group 2. (T_3) - (1), (3), (4), (5), (7), (10).

Each group is treated as a single-terminal problem and solved in steps 2 through 11. The starting distance matrix for each group is shown in Table 3.3 and Table 3.7. At this point, group index i is set equal to 1.

Step 2. Construct the initial minimum 1-tree for the i-th group of demand points and associated terminal using the corresponding distance matrix. First a minimum spanning tree is constructed over the demand points (or nodes) and two minimum arcs are drawn from the terminal to the minimum spanning tree. The minimum spanning tree is constructed by taking the minimum distances from the distance matrix. It is important that there is no loop formed among the subset or complete set of nodes. In construction of the minimum 1-tree, if there are arcs which are mutually exclusive for the membership of minimum 1-tree but have the same costs of travel, form more than one minimum 1-tree differing from each other by corresponding candidate arcs. For our sample problem, the associated minimum 1-tree for

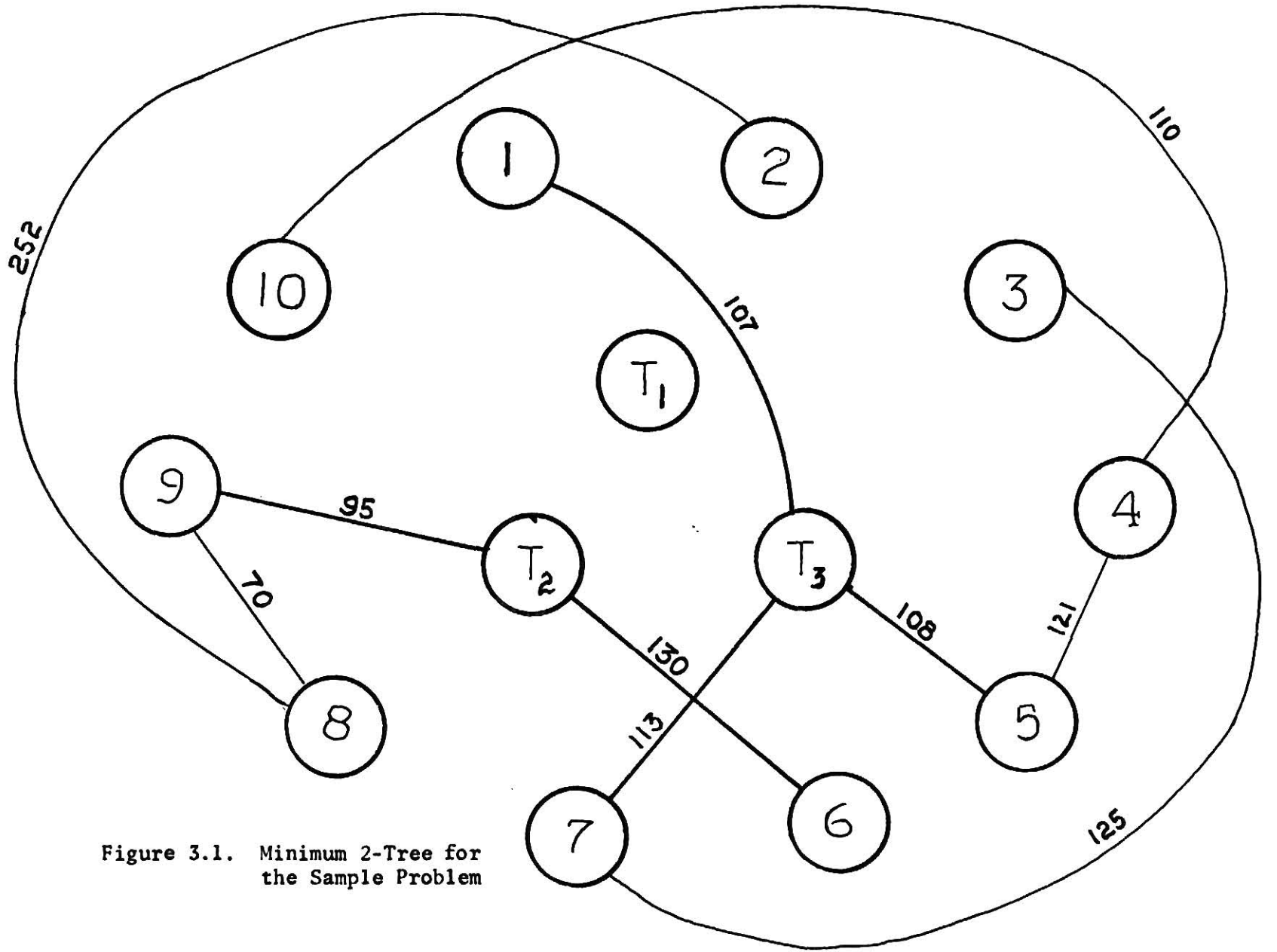


Figure 3.1. Minimum 2-Tree for the Sample Problem

the first group of demand points is shown in Figure 3.2. For the first group (the nodes with terminal (2)), the minimum spanning tree is constructed over nodes (2), (6), (8), and (9). Then two minimum arcs are drawn from (T_2). The graph becomes a minimum 1-tree. The algorithm is initiated by setting level L = 0 for the minimum 1-tree. The lower bound on the distance of tour for the i-th group is computed at level 0 such that

$$C_i^0 = \sum_{(k,\ell)} d(k,\ell), \qquad (k,\ell) \in A_i^0,$$

where index i refers to individual group and A_i^0 is the corresponding minimum 1-tree at level 0. Hence,

$$C_1^0 = 252 + 200 + 70 + 130 + 95$$

$$= 747$$

Step 3. Identify out-of-kilter nodes. If there is only one minimum 1-tree for each group, identify all out-of-kilter nodes. But if there are more than one minimum 1-tree and three is at least one node (k) which is *consistently* out-of-kilter, then the particular node is chosen for computing cost increment. The third possibility at this stage could be of the number of minimum 1-trees and there is no consistently out-of-kilter node. For each minimum 1-tree, out-of-kilter nodes are identified and corresponding increments are computed in step 4. Figure 3.2 depicting the initial minimum 1-tree for group 1 shows that for group 1, node (2) is out-of-kilter low and node (9) is out-of-kilter high.

Step 4. Compute the cost increments. If node (k) is out-of-kilter

Table 3.3 Distance Matrix for the First Group with Terminal T_2.

6	272			
8	252	250		
9	290	200	70	
T2	336	130	170	95
Nodes ↳	2	6	8	9

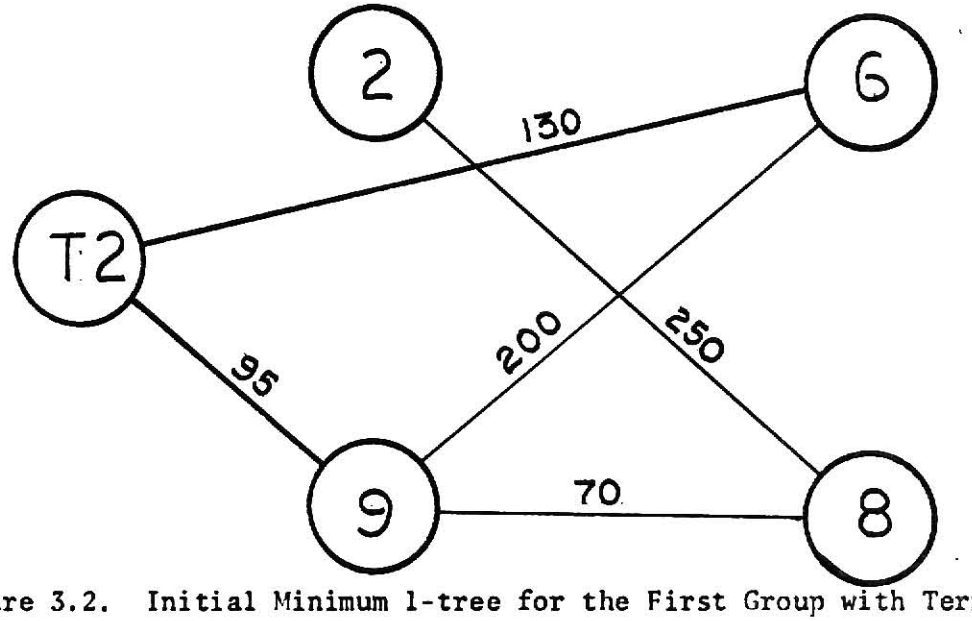


Figure 3.2. Initial Minimum 1-tree for the First Group with Terminal T_2.

low, the increase in the cost of the minimum 1-tree is computed such that

$$\delta_i^L = \min_{(k,\ell)\notin A_i^L} [d(k,\ell) - \max_{(r,s)\in A_i^L} [d(r,s)]], \quad (k) \neq (r), (s),$$

where $\{(r,s)\}$ is a set of arcs which can possibly be replaced by arc (k,ℓ) without forming a loop among nodes 1,2, ..., n but forms a loop with node (T_i). For group 1, node (2) is out-of-kilter low. The increase in the distance of the minimum 1-tree is computed such that

$$\delta_1^0(2) = \min_{(2,\ell)\notin A_1^0} [d(2,\ell) - \max_{(r,s)\in A_1^0} [d(r,s)]]$$

$$= \min \begin{bmatrix} d(2,6) - \max \begin{pmatrix} d(6,9) \\ d(8,9) \end{pmatrix} \\ d(2,9) - \max [d(8,9)] \\ d(2,T_2) - \max \begin{pmatrix} d(6,T_2) \\ d(9,T_2) \end{pmatrix} \end{bmatrix}$$

$$= \min \begin{bmatrix} 272 - \max \begin{pmatrix} 200 \\ 70 \end{pmatrix} \\ 290 - 70 \\ 336 - \max \begin{pmatrix} 130 \\ 95 \end{pmatrix} \end{bmatrix}$$

$$= \min \begin{bmatrix} 272 - 200 \\ 290 - 70 \\ 336 - 130 \end{bmatrix} = \min \begin{bmatrix} 72 \\ 220 \\ 206 \end{bmatrix}$$

$$= 72.$$

However, if node (k) is out-of-kilter high, the increase in the cost of the minimum 1-tree is computed such that

$$\delta_i^L(k) = \min_{(k,\ell)\in A_i^L} \left[\min_{(r,s)\notin A_i^L} [d(r,s)] - d(k,\ell) \right], \quad (k) \neq (r),(s)$$

where $\{(r,s)\}$ is a set of arcs which can possibly replace arc (k,ℓ) without forming a loop among nodes 1,2, ..., n_i of i-th group out forms a loop with node (T_i). In group 1, node (9) is out-of-kilter high at level 0 and the increase in the cost of the corresponding minimum 1-tree is computed such that,

$$\delta_1^0(9) = \min_{(9,\ell)\in A_1^0} \left[\min_{(r,s)\in A_1^0} [d(r,s)] - d(9,\ell) \right]$$

$$= \min \begin{pmatrix} \min \begin{pmatrix} d(2,6) \\ d(6,8) \end{pmatrix} - d(9,6) \\ \min \begin{pmatrix} d(2,6) \\ d(6,8) \end{pmatrix} - d(9,8) \\ \min \begin{pmatrix} d(2,T_2) \\ d(8,T_2) \end{pmatrix} - d(9,T2) \end{pmatrix}$$

$$= \min \begin{pmatrix} \min \begin{pmatrix} 272 \\ 250 \end{pmatrix} - 200 \\ \min \begin{pmatrix} 272 \\ 250 \end{pmatrix} - 70 \\ \min \begin{pmatrix} 336 \\ 170 \end{pmatrix} - 95 \end{pmatrix} = \min \begin{pmatrix} 250 - 200 \\ 250 - 70 \\ 170 - 95 \end{pmatrix}$$

$$= \min \begin{pmatrix} 50 \\ 180 \\ 75 \end{pmatrix} = 50$$

Step 5. Select a node having the minimum cost increment for each group. For group 1, nodes (2) and (9) are out-of-kilter and the increments due to each is computed in step 4. Node (9) is selected as the node which

yields the minimum increment

$$\delta_1^*(k) = \min_{(k)} [\delta_1^0(k)], \qquad k = 2, 9$$

or

$$\delta_1^*(9) = 50.$$

After finding $\delta_1^*(k)$, level L in increased by one and the lower bound on the cost of tour is updated as

$$C_i^L = C_i^{L-1} + \delta_i^*(k)$$

$$C_1^1 = C_1^0 + \delta_1^*(k)$$

$$= 747 + 50$$

$$= 797$$

For any group at any level if a tie exits for the minimum $\delta_i^*(k)$, a node is selected by any particular rule.

Step 6. Update the distance matrix and associated minimum 1-tree(s). If the selected node (k) is out-of-kilter low, the distance matrix is updated such that

$$\hat{\delta}_i(k,\ell) = d_i(k,\ell) - \delta_i^*(k), \ (\ell) = 1,2, \ldots, n, T_i; \ (\ell) \neq (k),$$

and consequently, a minimum 1-tree(s) is constructed from the updated distance matrix. However, if the selected node (k) is out-of-kilter high, the distance matrix is updated as

$$\hat{\delta}_i(k,\ell) = d_i(k,\ell) + \delta_i^*(k), \ (\ell) = 1,2, \ldots, n_i, T_i; \ (\ell) \neq (k),$$

and a minimum 1-tree(s) is constructed from the updated distance matrix.

As shown earlier, node (9) is out-of-kilter high with an increment of $\delta_i^*(9)$ or 50 for group 1. The updated distance matrix and corresponding two minimum 1-trees are shown in Table 3.4 and Figure 3.3, respectively. At level 1, two minimum 1-trees are formed as there exists a tie between arcs (6,8) and (6,9) and both the arcs are mutually exclusive as far as their inclusion in the minimum 1-tree is concerned. If both were present simultaneously, a loop is formed among nodes (6), (8), and (9) against the definition of the minimum 1-tree. The two minimum 1-trees differ from each other depending on the candidate arcs (6,9) and (6,8).

Step 7. Check for the minimum tour for each group. If at the particular level all nodes in any minimum 1-tree are in-kilter, the tour is identified and corresponding cost is $C_i^* = C_i^L$. However, if no minimum 1-tree forms a tour, steps 2 through 6 are repeated. At level 1, for first group, the minimum 1-trees do not form tours yet and steps 2 through 6 are repeated till any minimum 1-tree becomes a tour. At level 2 and for group 1, the following tour is identified from the minimum 1-tree.

$$T_2 - 6 - 2 - 8 - 9 - T_2,$$

with the total distance of 819. The updated distance matrix for level 2 and associated minimum 1-tree are shown in Table 3.5 and Figure 3.4. The summary of results for group 1 is shown in Table 3.6.

Step 8. Decompose each minimum tour into subtours. The increments in the distance of each minimum tour, resulting due to the cut at a particular arc of the minimum tour, is computed such that

$$\Delta_i(k,\ell) = d(k, T_i) + d(\ell, T_i) - d(k,\ell).$$

Table 3.4 Updated Distance Matrix for First Level for the First Group with Terminal T_2.

6	272			
8	252	250		
9	340	250	120	
T2	336	130	170	145
Nodes	2	6	8	9

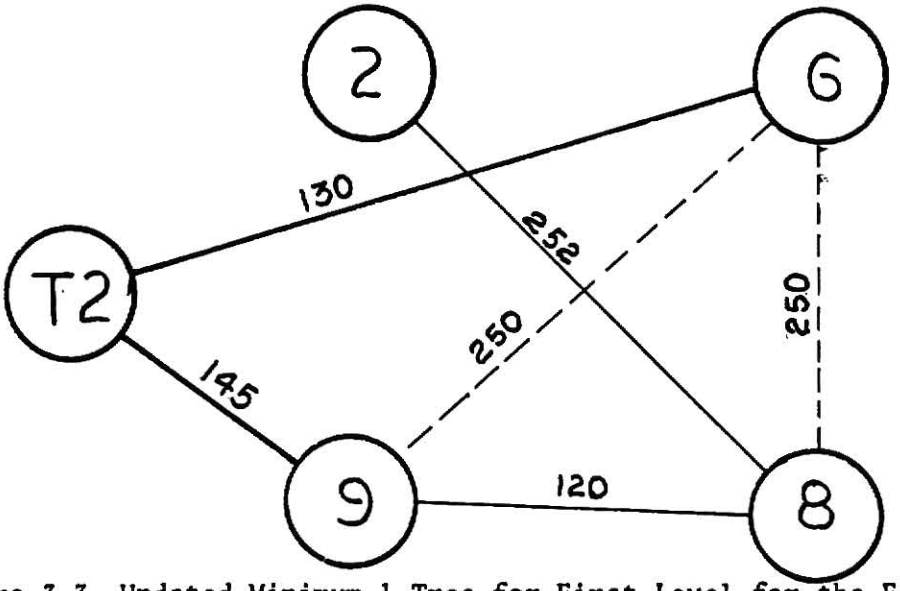


Figure 3.3 Updated Minimum 1-Tree for First Level for the First Group with Terminal T_2.

Table 3.5 Updated Distance Matrix for Second Level for the First Group with Terminal T_2.

6	250			
8	232	250		
9	318	250	120	
T2	314	130	170	145
Nodes →	2	6	8	9

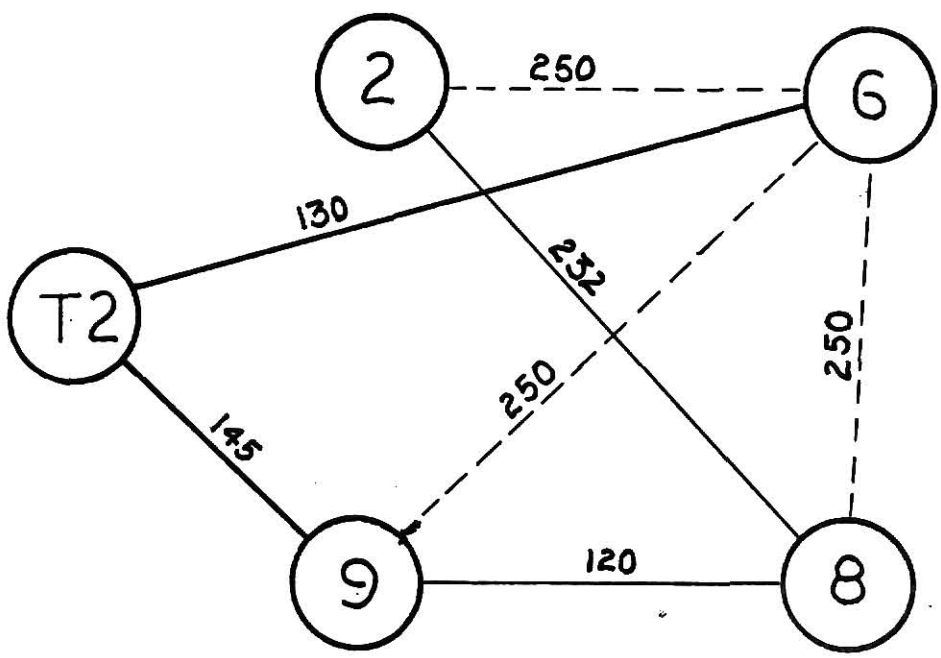


Figure 3.4 Updated Minimum 1-Tree for Second Level for the First Group with Terminal T_2.

Table 3.6 Summary of Results for Minimum Tour of First Group

Level L	Number of Minimum 1-Trees	Candidates Arcs	Arcs Comprising the Minimum 1-Tree $(k,\ell) \in A_1^L$	Out-of-Kilter Nodes: High Node (k)	Out-of-Kilter Nodes: High Cost Increment $\delta_1^L(k)$	Out-of-Kilter Nodes: Low Node (k)	Out-of-Kilter Nodes: Low Cost Increment $\delta_1^L(k)$	Minimum Cost Increment $\delta_1^*(k)$	Selected Node (k^*)	Lower Bound on the Distance of Tour C_1^L
0	1		(T2,6),(T2,9),(2,8) (6,9),(8,9)	9	50	2	72	50	9	747
1	2	(6,9)	(T2,6),(T2,9),(2,8) (6,9),(8,9)			2	22	22	2	797
		(6,8)	(T2,6),(T2,9),(2,8) (6,8),(8,9)							
2	3	(6,9)	(T2,6),(T2,9),(2,8) (6,9),(8,9)							819
		(6,8)	(T2,6),(T2,9),(2,8) (6,8),(8,9)							
		(2,6)	T2-6-2-8-9-T2**							

** Minimum 1-tree is a tour.

At this point a counter is initiated as K = 0. As the sum of demands associated with nodes in group 1 is within the truck capacity, the minimum tour of this group is the feasible tour.

Step 9. Check for feasible subtours. The arc with the minimum value of increment at which the cut is made is selected according to the following conditions: (1) feasible subtours are formed, and (2) demand points are evenly distributed in each subtour. If neither of the conditions is satisfied, the cut with the next minimum increment is selected and checked for the above two conditions. The procedure is repeated till a cut is found. The corresponding lower bound on the cost of all subtours is computed as

$$C_i^* = C_i^* + \Delta_i(k,\ell)$$

Step 10. Identify the subtours. If no feasible subtour is formed by certain cut, go to step 8 for the non-feasible subtours formed by the cut and each non-feasible subtour is treated as a tour. If a feasible subtour (or subtours) is formed, eliminate the nodes from further consideration and counter K is updated by the number of nodes in the feasible subtour(s). If there is any non-feasible subtour, go to step 9. The counter K is updated such that

$$\begin{aligned} K &= K + N[\Delta_i(k,\ell)] \\ &= 0 + N\ [\Delta_i(k,\ell)] \\ &= 0 + 4 = 4, \end{aligned}$$

where again $N[\Delta_i(k,\ell)]$ is the number of nodes forming feasible subtour(s) when cut at (k,ℓ) in the i-th group is considered.

Step 11. Check for all subtours. If counter K = n_i, where n_i is the total number of demand points in the i-th group, the solution to the i-th delivery subproblem is obtained and all feasible subtours are identified. Otherwise, steps 8 through 10 are repeated for all nonfeasible subtours.

Step 12. Check for all groups. If index i is equal to p all the groups are considered and the solution to the original problem is obtained. At this point all subtours, their respective loads and distances are identified. If index i is not equal to p, steps 2 through 11 are repeated for the next group. This is done by updating index i to i+1 and carrying out steps 2 through 11. In our sample problem, the value of p is 2. After finding the minimum subtour for group 1, index i is increased by 1 and steps 2 through 11 are repeated for group 2. For group 2, the initial distance matrix with terminal 3 and initial minimum 1-tree are shown in Table 3.7 and Figure 3.5, respectively. Steps 2 through 7 are summarized in Table 3.8. The minimum tour for this group is identified as T_3 - 7 - 3 - 5 - 4 - 10 - 1 - T_3 with the total distance of 875. Having identified the minimum tour, it is considered for cuts. The cuts and associated increments are shown in Table 3.9. Thus, steps 8 through 11 are repeated for group 2 resulting in two feasible subtours. The subtours are T_3 - 1 - 10 - 4 - T_3 and T_3 - 5 - 3 - 7 - T_3 with distnaces of 503 and 513 respectively. As the value of p is 2 (that is, there are only two groups) the final solution is reached and the feasible subtours, the corresponding loads and distances are displayed in Table 3.11. Table 3.10 shows the truck availability and allocation.

Table 3.7 Distance Matrix for the Second Group with Terminal T_3.

3	331					
4	152	284				
5	172	167	121			
7	217	125	216	114		
10	132	388	110	227	301	
T3	107	224	154	108	113	207
Nodes	1	3	4	5	7	10

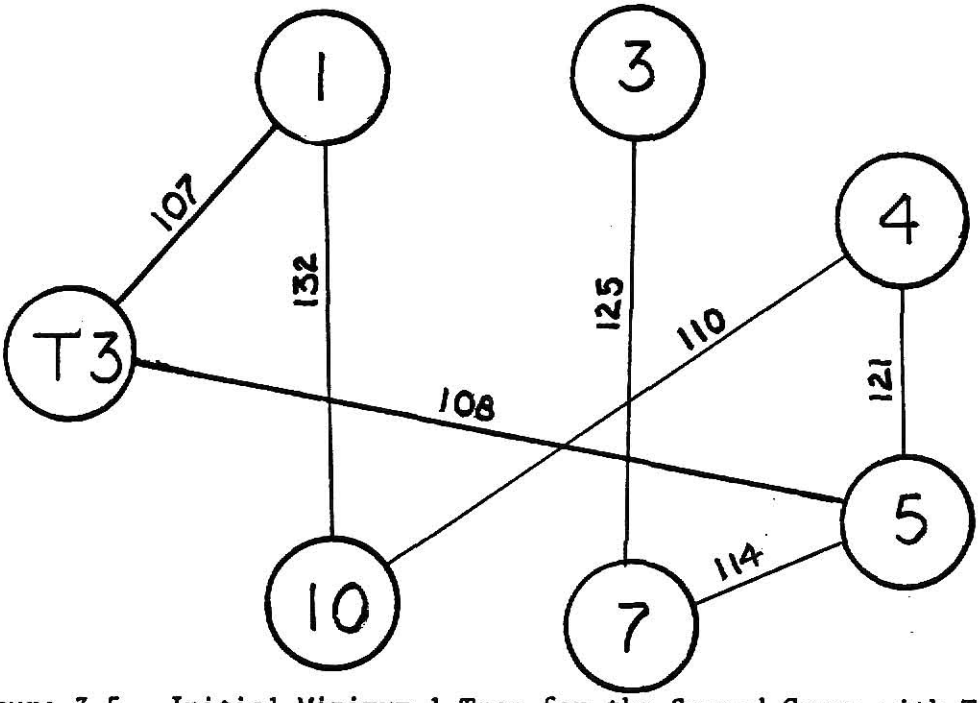


Figure 3.5. Initial Minimum 1-Tree for the Second Group with Terminal T_3.

Table 3.8 Summary of Results for Minimum tour of Second Group.

Level	Number of Minimum 1-Trees	Candidate Arcs	Arcs Comprising the Minimum 1-Tree	Out-of-Kilter Nodes				Minimum Cost Increment	Selected Node	Lower Bound on the Distance of Tour
				High		Low				
				Node	Cost Increment	Node	Cost Increment			
L			$(k,\ell) \in A_2^L$	(k)	$\delta_2^L(k)$	(k)	$\delta_2^L(k)$	$\delta_2^*(k)$	(k^*)	C_2^L
0	1		(T3,1),(T3,5),(1,10) (3,7),(4,5),(4,10), (5,7)	5	5	3	53	5	5	817
1	2	(5,T3)	(T3,1),(T3,5),(1,10) (3,7),(4,5),(4,10), (5,7)			3	53	53	3	822
		(7,T3)	(T3,1),(T3,7),(1,10) (3,7),(4,5),(4,10), (5,7)			3	53	53	3	822
2	4	(5,T3) (5,7)	(T3,1),(T3,5),(1,10) (3,7),(4,5),(4,10), (5,7)							
		(5,T3) (3,5)	(T3,1),(T3,5),(1,10) (3,7),(4,5),(4,10), (3,5)							875
		(7,T3) (5,7)	(T3,1),(T3,7),(1,10) (3,7),(4,5),(4,10), (5,7)							
		(7,T3) (3,5)	T3-7-3-5-4-10-1-T3**							

** Minimum 1-tree is a tour.

Table 3.9 Possible Increments Resulting Due to Cut at a Particular Arc.

No.	Arc for the Cut (k,ℓ)	Increase Due to Cut $\Delta_i(k,\ell)$	Bound on all Subtours ω_i^*
1	(1,10)	182	
2	(10,4)	251	
3	(4,5)	141*	875
4	(5,3)	275	
5	(3,7)	212	

*arc chosen for cut

Table 3.10 Capacity and Availability of Trucks at End of First Cut.

Truck	40	30	20
Available	1	3	10
Allocated	1	2	0

Table 3.11 Table Showing Final Solution to the Sample Delivery Problem.

No.	Terminal	Route	Load	Distance
1	2	T_2 - 6 - 2 - 8 - 9 - T_2	29	819
2	3	T_3 - 1 - 10 - 4 - T_3	35	503
3	3	T_3 - 5 - 3 - 7 - T_3	28	513

3.3 Computational Algorithm

The following is a step by step computational algorithm for solving the multi-terminal delivery problem.

Step 1: Construct the minimum p-tree.

1.1 Identify the groups of demand points for respective terminals and construct the respective distance matrices.

1.2 Let group index $i = 1$.

Step 2: Construct the initial minimum 1-tree for the i-th group.

2.1 Set level $L = 0$.

2.2 Draw a minimum spanning tree(s) from which a minimum 1-tree(s) is constructed with cycle around node (T_i).

2.3 Compute the corresponding lower bound on minimum tour cost such that

$$C_i^L = \sum_{(k,\ell)} d(k,\ell), \qquad (k,\ell) \in A_i^L$$

Step 3: Identify out-of-kilter nodes for minimum 1-tree(s) of the i-th group.

3.1 If there is only one minimum 1-tree, identify all out-of-kilter nodes, and go to step 4.

3.2 If there are more than one minimum 1-tree and there is at least one node (k) which is consistently out-of-kilter, go to step 4 to compute the cost increment for that node only.

3.3 If there are more than one minimum 1-tree and there is no consistently out-of-kilter node, identify out-of-kilter nodes for each minimum 1-tree and go to step 4 to compute the corresponding cost increments.

Step 4: Compute cost increments in the minimum 1-tree of i-th group.

4.1 If node (k) is out-of-kilter low, compute the increase in the cost of minimum 1-tree such that

$$\delta_i^L(k) = \min_{(k,\ell)\notin A_i^L} [d(k,\ell) - \max_{(r,s)\in A_i^L} [d(r,s)]], \quad (k) \neq (r),(s)$$

where $\{(r,s)\}$ is a set of arcs which can possibly be replaced by arc (k,ℓ) without forming a loop among nodes $1, 2, \ldots, n_i$ but forms a loop with node (T_i).

4.2 If node (k) is out-of-kilter high, compute the increase in the cost of minimum 1-tree such that

$$\delta_i^L(k) = \min_{(k,\ell)\in A_i^L} [\min_{(r,s)\notin A_i^L} [d(r,s)] - d(k,\ell)], \quad (k) \neq (r),(s)$$

where $\{(r,s)\}$ is a set of arcs which can possibly replace arc (k,ℓ) without forming a loop among nodes $1,2, \ldots, n_i$ but forms a loop with node (T_i).

Step 5: Select a node having the minimum cost increment for the i-th group.

5.1 Find the minimum increment such that

$$\delta_i^*(k) = \min_{(k)} [\delta_i^L(k)].$$

If a tie exists select a node by any particular rule.

5.2 Let level L = L+1 and update the lower bound on the cost of tour such that

$$C_i^L = C_i^{L-1} + \delta_i^*(k).$$

Step 6: Update the distance matrix and associated minimum 1-tree(s) for the i-th group.

6.1 If the selected node (k) is out-of-kilter low, update the distance matrix such that

$$\hat{\delta}(k,\ell) = d(k,\ell) - \delta_i^*(k),\ (\ell) = 1,2,\ \ldots,\ n_i,\ T_i;\ (\ell) \neq (k)$$

and construct the corresponding minimum 1-tree(s). Then, go to step 7.

6.2 If the selected node (k) is out-of-kilter high, update the distance matrix such that

$$\hat{d}(k,\ell) = d(k,\ell) + \delta_i^*(k),\ (\ell) = 1,2,\ \ldots,\ n_i,\ T_i;\ (\ell) \neq (k)$$

and construct the corresponding minimum 1-tree.

Step 7: Check for minimum tour for the i-th group.

7.1 If all nodes in any minimum 1-tree are in-kilter, identify the tour and corresponding cost such that

$$C_i^* = C_i^L.$$

7.2 If minimum tours are obtained for all the groups of demand points go to step 8.

7.3 If no minimum 1-tree consists of all in-kilter nodes, go to step 3.

Step 8: Decompose the minimum tour into subtours for the i-th group.

8.1 If minimum tour of a group is a feasible tour, identify the corresponding route.

8.2 Compute the increments in the distance of minimum tour due to cut at particular arc such that

$$\Delta_i(k,\ell) = d(k,\ T_i) + d(\ell,\ T_i) - d(k,\ell)$$

8.2 Compute the increments in the distance of the minimum tour due to cut at particular arc such that

$$\Delta_i(k,\ell) = d(k, T_i) + d(\ell, T_i) - d(k,\ell).$$

8.3 Initiate counter K = 0.

Step 9: Check for feasible subtours for the i-th group.

9.1 Select an arc with the minimum value of increment at which the cut is made according to the following conditions: (1) Feasible subtours are formed, and (2) demand points are evenly distributed in each subtour. If neither of the conditions is satisfied, the cut with the next minimum increment is selected and checked for the above two conditions. The procedure is repeated till a cut is found.

9.2 Compute the lower bound on the cost of all subtours of the group such that

$$C_i^* = C_i^* + \Delta_i(k,\ell)$$

Step 10. Identify the subtours for the i-th group.

10.1 If any feasible subtour is formed, identify the subtour and eliminate the nodes of the feasible tour from consideration.

10.2 Update the counter such that

$$K = K + N[\Delta_i(k,\ell)]$$

where $N[\Delta_i(k,\ell)]$ is the number of nodes forming a feasible subtour when the cut at (k,ℓ) is considered.

Step 11. Check for all subtours.

11.1 If $K = n_i$, identify all feasible subtours.

11.2 If $K < n_i$, go to step 9 treating non-feasible subtour as a tour and consider it for cuts.

Step 12. Check for all groups.

12.1 If $i = p$, the solution is obtained and identify all the subtours, respective loads, distances and the total distance of all subtours.

12.2 If $i < p$, let $i = i + 1$ and go to step 2.

CHAPTER IV

COMPUTATIONAL EXPERIMENTS

This chapter is devoted to investigating the computational experience with the graph-theoretic algorithm for delivery problems, as discussed in preceding chapters. The multi-terminal algorithm has been coded in FORTRAN IV for the IBM 360/50 computer. The same program can be used to solve single-terminal problems by specifying the number of terminals equal to one. The program listing is displayed in Appendix A. The problems tried, ranged in size from 1 terminal to 5 terminals and 6 demand points to 25 demand points. Most of the problems are taken from Herring [12].

The performance of the algorithm has been evaluated with respect to two measures: (1) the quality of solution; and (2) the computational time. The results obtained by the proposed algorithm have been divided in two classes: (1) single-terminal delivery problems,and (2) multi-terminal delivery problems.

The analysis of results shows the following significant observations for the respective classes.

4.1 Single-terminal delivery problems:

The merit of the graph-theoretic approach has been studied by solving a wide range of problems. The size of the single-terminal delivery problems vary with respect to the number of demand points. A total of 26 problems have been solved as shown in Table 4.1. The data and corresponding solutions with final routes, corresponding loads and distances, total distance of travel and final truck availability and allocation

table for each single-terminal delivery problem are displayed in Appendix B.

Quality of solution. The solution obtained by the graph-theoretic algorithm are compared with the corresponding best solutions obtained by savings approach. However, the solutions obtained by Clarke and Wright [4], Tillman and Cochran [17] and Herring [12] are displayed in Table 4.2. Table 4.1 displays the solutions obtained by the proposed algorithm and by savings approach. The scarcity in the number of available solutions limits the comparison criterion. However, it can be observed that the solutions by graph-Theoretic approach, except for problems 23 and 24, are the same as or superior to the best solutions by savings approach.

Computational time. The computational time is of course, one of the main considerations in the evaluation of the performance of the proposed graph-Theoretic algorithm. Average times for the same size problems are shown in Table 4.3. Figure 4.1 shows a plot of the average computer time against varying demand points. Points are joined by straight lines because this is a case of discrete variables. As can be noticed, there is not much variation in the computational time, when problems having 6, 7 and 8 demand points are solved. However, the computational time increases very rapidly with the increase in the number of demand points. At this stage of research, only a speculative explanation can be made concerning the rapid increase in the computational time when the number of demand points increases. The increase in computational time can be attributed to the fact that in the construction of the minimum 1-tree(s), the search is conducted, at each stage, in the updated distance matrix. The other factor is the computation of minimum cost increment to make out-of-kilter nodes in-kilter nodes.

Table 4.1 Summary of Solutions of Single-Terminal Delivery Problems.

Problem No.	Number of Demand Points	Best Solution		Computational Time (min)
		Saving Method	Proposed Method	
1	6	*	1016	0.92
2	6	394	394	1.02
3	6	*	337	0.91
4	6	*	306	0.55
5	6	*	1808	2.77
6	6	*	974	0.78
7	6	*	2260	1.02
8	7	*	1193	0.97
9	8	*	2228	0.90
10	9	*	918	8.25
11	9	*	1808	2.84
12	10	76	76	10.62
13	10	2470	2470	9.6
14	10	2156	2156	4.21
15	10	1679	1679	5.45
16	10	4109	3941	1.40
17	10	*	2626	12.25
18	10	*	2550	7.9
19	10	*	3102	12.5
20	10	*	691	12.2
21	14	*	1141	15.6
22	25	9604	9553	30.72
23	25	8325	8582	61.44
24	25	2289	2295	25.56
25	25	2638	2638	36.12
26	25	8901	8901	25.66

*These problems have not been solved by savings approach.

Table 4.2 Summary of distances for single-terminal problem

Problem No.	Number of Demand Points	Clarke & Wright Method	Tillman & Cochran's 2-Decision Look-Ahead Method			Herring's 3-Decision Look-Ahead Method			Proposed Method
			2 Alternatives	3 Alternatives	5 Alternatives	2 Alternatives	3 Alternatives	5 Alternatives	
12	10	76	76	76	76	76	76	76	76
13	10	2846	2470	2470	2470	2470	2470	2470	2470
14	10	2156	2156	2156	2156	2156	2156	2156	2156
15	10	1813	1679	1679	1679	1679	1679	1679	1679
16	10	4146	4146	4121	4121	4146	4109	4109	3941
22	25	9604	9990	9848	9848	9709	9689	9689	9553
23	25	8383	8325	8325	8325	8325	8325	8325	8582
24	25	2289	2289	2289	2289	2289	2289	2289	2295
25	25	2788	2788	2788	2788	2657	2638	2638	2638
26	25	8901	8901	8901	8901	8901	8901	8901	8901

Table 4.3 Average Computational Times for Various Sets of Single-Terminal Delivery Problems.

Set No.	Number of Demand Points	Number of Problems	Average Computational time (min)
1	6	7	1.12
2	7	1	0.97
3	8	1	0.90
4	9	2	5.55
5	10	9	7.62
6	14	1	15.6
7	25	5	35.9

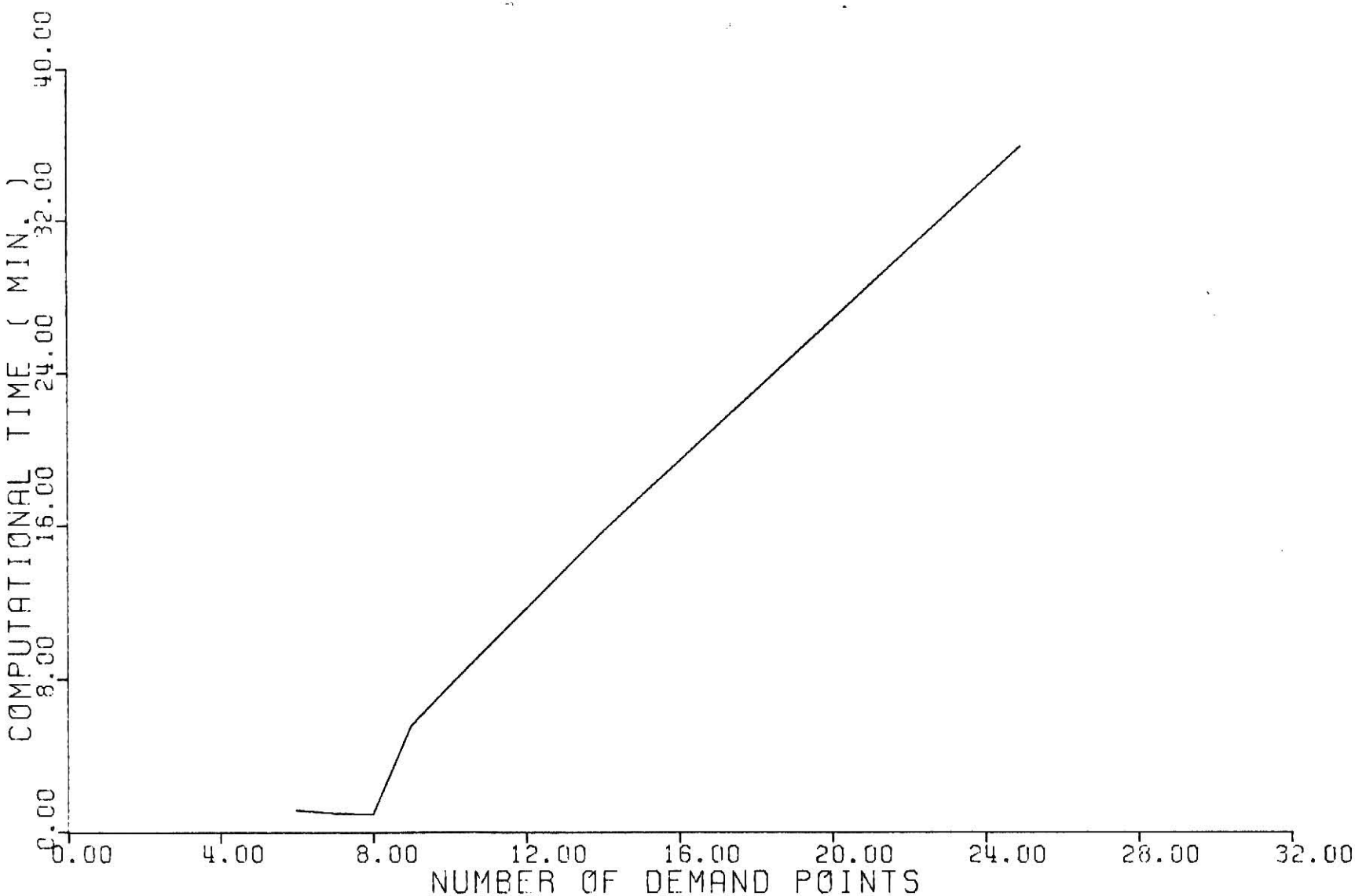


Fig. 4.1 Average Computer Time Versus Number of Demand Points

4.2 Multi-terminal Delivery Problems:

As mentioned previously, two sets of problems have been solved. The first set consists of five problems with three terminals and ten demand points. The second set comprises of five problems with five terminals and twenty five demand points. Both sets of problems are taken from Herring [12]. The data of both sets of problems and corresponding solutions including final routes, respective loads, distances, the total distance of travel, and final truck allocation are displayed in Appendix C. As mentioned earlier, the measures of performance including the quality of solution and the computational time are considered for comparison purpose.

Quality of solutions. The summary of the solutions obtained for all multi-terminal delivery problems is shown in Table 4.4. It can easily be noticed that for small sized problems the proposed algorithm has produced solutions similar or superior to those obtained by the savings approach. As far as five terminals and twenty-five demand points problems are concerned, the proposed method does not produce as good results as those produced by savings approach. However, the difference is not very significant.

Computational time. Observing Table 4.4 for the computational time, it can be easily noticed that the time increases very rapidly with the increase in the number of terminals and number of demand points. However, due to lack of data for comparison no comment can be made regarding the relationship between the computational time and the number of terminals. The major contributors to the computational time are three important and main features of the multi-terminal algorithm. First is the construction of minimum p-tree to assign various demand points to various terminals. In order to form minimum p-tree a search is to be made through the distance

Table 4.4. Summary of Solutions of Multi-terminal Delivery Problems.

Problem Number	Number of Terminals	Number of Demand Points	Savings Approach	Proposed Approach	Efficiency of Solution*	Computational Time (min)	Average Computation Time (min)	Average Efficiency
1	3	10	59	59	100	1.68		
2	3	10	1835	1835	100	1.42		
3	3	10	2042	1857	110	1.34	1.47	103.5
4	3	10	1706	1638	104	1.38		
5	3	10	3490	3468	103.5	1.52		
6	5	25	6832	6857	99.9	25.35		
7	5	25	6099	6695	91.0	21.38		
8	5	25	1377	1470	94.0	13.71	19.39	91.03
9	5	25	1722	2236	77.5	25.85		
10	5	25	6653	6915	96.75	10.49		

*Efficiency of the solution is defined as the quotient of solution by savings approach to that by proposed method.

matrix for the minimum distance elements. Second is the construction of the minimum 1-tree for each group of demand points. Third is the computation of cost increment in the cost of the minimum tour to convert out-of-kilter nodes to in-kilter nodes for each group of demand points at each level.

CHAPTER V

SUMMARY AND CONCLUSIONS

Delivery problems are of frequent occurrence in day-to-day life with a seemingly endless variety of characteristics and they arise in an innumerable variety of contexts. The truck routing problem in its simplest form is to determine the routes such that the distance travelled is a minimum and the following conditions are satisfied: (1) all demands are met; (2) no carrier is loaded with more than its capacity; (3) only the available number of trucks are used.

In practice, the above problem may be complicated by one or more added characteristics. For instance, there may be a prespecified due-date for delivery at particular demand points, and/or delivery can not be made before some specified time. Other type of characteristics can be that of occurrence of practical difficulties such as these imposed by labor union. The driver should not drive more than certain number of hours everyday. delivery at particular demand points and/or delivery can not be made beconsiderably and these new types of problems are referred to as a multiterminal delivery problem. In the above discussion, the demands are assumed to be known. A new category of carrier routing problems evolves when demands are of probabilistic nature.

The delivery problem may be regarded as a generalization of the classical traveling salesman problem. Although literature on traveling salesman problem is in abundance, very little can be found which directly relates to the carrier routing problem. The principal methods for solving

the single-terminal delivery problem are dynamic programming (Tillman, 1965; Gonzales, 1962; and Held and Karp, 1970), branch-and-bound (Little et. al, 1963; and Pierce, 1969) integer programming (Miller, Tucker and Lemlin, 1960; Balinski and Quandt, 1964) and finally heuristic programming (Dantzig and Ramser, 1959; Clarke and Wright, 1964; Cochran, 1967; Hering, 1970). The methods available for multiterminal delivery problems are heuristics (Tillman, 1969; and Hering, 1970).

The purpose of this chapter is to briefly discuss the heuristic algorithm that exploits the underlying combinatorial nature of the problem, solve a simple multi-terminal illustrative example and report on the merit of such algorithm.

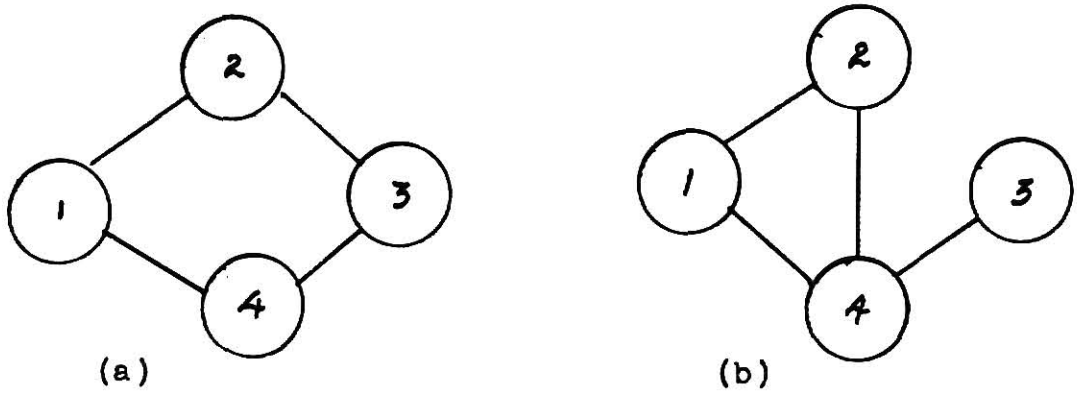
5.1 Development of an Out-of-Kilter Algorithm.

The graph-theoretic algorithm to solve multi-terminal delivery problems consists of three main phases. Phase I assigns the various demand points to terminals, the distance being the decision criterion. Phase II finds the minimum tours for each group of demand points including the associated terminal. Phase III decomposes the obtained minimum tours into subtours to satisfy the various constraints such as the capacity and number of carriers.

Central to the approach of associating the demand points with a particular terminal is the concept of minimum p-tree. A p-tree is a graph over nodes (k), $k = 1, 2, \ldots, n$ and $i_1, i_2, \ldots, i_p$ such that any node (k) is associated with only one of i_r's and there is no loop among the set of these nodes. A node i_r is referred to as an origin. A p-tree with minimum cost is referred to as a minimum p-tree. In case of multi-terminal delivery problems, the terminals are the origins.

The maximum integer value p of the p-tree which can take, is the number of terminals. After assigning the demand points to a particular terminal, each group of demand points associated with each terminal is treated as a single-terminal delivery problem with common constraints of carrier capacity and available number of carriers and is solved as follows.

Phase II of the graph-theoretic algorithm is to find the minimum tour from a minimum 1-tree for each group of demand points. To distinguish between tour and a 1-tree, consider the following two connected graphs with same nodes and same total number of arcs.



Both graphs have the following salient features: (1) the nodes represent cities or demand points; and (2) there are four connecting arcs. But they differ from each other in one respect and that being the number of arcs, emanating from each node. In (a) each node has the same number of arcs, that is, 2 and the connecting graph represents a tour. The arcs associated with nodes distinguish between the two graphs. Graph (b) is termed as 1-tree. A 1-tree can be formally defined as a connected graph without cycles on nodes 1,2, ..., k-1, k+1, ..., n with two arcs incident with node (k). Thus from graphs (a) and (b), a tour is a 1-tree with each of its nodes having two arcs (or 2 degrees). A degree of a

node is the number of arcs from the node. A node with two degrees is said to be in-kilter. Consequently, any node which does not have two degrees is out-of-kilter. Depending on whether the node has more or less than 2 degrees, it is identified as out-of-kilter high or out-of-kilter low node, respectively. From graph (a) and (b), it is evident that every tour is a 1-tree but the reverse is not always true. In most cases, the problem basically is of getting a tour from a 1-tree, or more specifically, to convert out-of-kilter nodes of 1-tree into in-kilter nodes.

Most of the time, interest is of getting a minimum or optimal tour. This can be achieved by first constructing a minimum 1-tree (1-tree with minimum cost) and then gradually converting the out-of-kilter nodes into in-kilter and taking care that each of such conversion is associated with a minimum possible increment in the cost of the minimum 1-tree. Also, it is evident that if a minimum 1-tree is a tour, it is the 'best' route and it is the optimal solution to the carrier routing problem given that there exists a carrier with capacity satisfying all demand points of the group in one tour.

The conversion of the minimum 1-tree to a minimum tour is made possible by the use of branch-and-bound approach. At each level L, out-of-kilter nodes are identified and the gap between the minimum 1-tree and the minimum tour is reduced by the amount

$$\delta^2(k) = \min\ [\bar{d}(k,\ell) - d(r,s)],$$

where, for out-of-kilter low node (k), arc (k,ℓ) is a substitute for

arc (r,s) which is already present in the minimum 1-tree. Similarly, for out-of-kilter high node (k), the amount by which the gap between the minimum 1-tree and the minimum tour reduces at each level is

$$\delta^{L}(k) = \min\ [\bar{d}(i,j) - d(r,k)],$$

where arc (i,j) is a substitute for (r,k) which is already present in the minimum 1-tree. The distance matrix is updated by the minimum $\delta^{L}(k)$ at each level L. That is, for the subsequent iterations, we use the modified distances, which are computed such that

$$\hat{d}(k,\ell) = d(k,\ell) + \delta^{L}(k),$$

where $\hat{d}(k,\ell)$ and $d(k,\ell)$ are the modified and original cost of travel from (k) to (ℓ), respectively. The increment $\delta^{L}(k)$ is either positive or negative depending on whether node (k) is out-of-kilter high or out-of-kilter low.

At any level, if there are arcs which are mutually exclusive for membership of the minimum 1-tree but have the same costs of travel, there exist more than one minimum 1-tree differing from each other by the corresponding candidate arcs. Branching is carried out at a particular arc if, at particular level, there is a number of minimum 1-trees and no node is "consistently out-of-kilter". An out-of-kilter node is said to be consistently out-of-kilter if it has the same degrees in all the minimum 1-trees at certain level. The bound on the cost of the minimum tour is found out as

$$C^{L} = C^{L-1} + \delta^{L-1}(k),$$

where

C^L = lower bound on cost of the minimum tour at level L

$\delta^{L-1}(k)$= cost increment resulting due to out-of-kilter node (k)

The initial lower bound at level 0 is

$$C^0 = \sum_{(k,\ell)} d(k,\ell), \qquad (k,\ell) \in A^L.$$

The lower bound associated with the final minimum 1-tree, which represents a tour, becomes the minimum distance of travel.

Having obtained the minimum tour, the constraints of the carrier capacity are applied. If a carrier capacity permits the visit to all demand points in one route, the minimum tour obtained is the solution and no further consideration is necessary. Otherwise, such a tour must be decomposed into feasible subtours to satisfy the constraints of the capacity and number of available carriers. Associated with the process of decomposition of minimum tour into feasible subtours is the concept of cut at a particular arc. A cut at arc (k,ℓ) is breaking of arc (k,ℓ) in minimum tour and connecting individual nodes (k) and (ℓ) to terminal (T). Increment in the distance of the tour is found out for the cut at each arc in the minimum tour. This can be formally stated as

$$\Delta(k,\ell) = d(T,k) + d(T,\ell) - d(k,\ell),$$

where $\Delta(k,\ell)$ is the increase in minimum route due to cut at (k,ℓ). The arc chosen for the cut depends on two considerations. The first consideration being whether the cut at an arc with the minimum increment produces feasible routes. The second consideration investigates whether

the cut distributes the demand points evenly in the sublinks. If any cut with minimum increase satisfies either of the above two conditions, it is then selected. Otherwise, the cut with the next minimum increase is chosen for consideration. The procedure is repeated with sublinks till the feasible subtours are formed. Now, the graph-theoretic algorithm for multi-terminal delivery problem can be stated in formal steps as follows:

Step 1: Construct the minimum p-tree.

1.1 Identify the groups of demand points for respective terminals and construct the respective distance matrices.

1.2 Let group index $i = 1$.

Step 2: Construct the initial minimum 1-tree for the i-th group.

2.1 Set level $L = 0$.

2.2 Draw a minimum spanning tree(s) from which a minimum 1-tree(s) is constructed with cycle around node (T_i).

2.3 Compute the corresponding lower bound on minimum tour cost such that

$$C_i^L = \sum_{(k,\ell)} d(k,\ell), \qquad (k,\ell) \in A_i^L.$$

Step 3: Identify out-of-kilter nodes for minimum 1-tree(s) of the i-th group.

3.1 If there is only one minimum 1-tree, identify all out-of-kilter nodes, and go to step 4.

3.2 If there are more than one minimum 1-tree and there is at least one node (k) which is consistently out-of-kilter, go to step 4 to compute the cost increment for that node only.

3.3 If there are more than one minimum 1-tree and there is no consistently out-of-kilter node, identify out-of-kilter nodes for each minimum 1-tree and go to step 4 to compute the corresponding cost increments.

Step 4: Compute cost increments in the minimum 1-tree of i-th group.

4.1 If node (k) is out-of-kilter low, compute the increase in the cost of minimum 1-tree such that

$$\delta_i^L(k) = \min_{(k,\ell)\notin A_i^L} [d(k,\ell) - \max_{(r,s)\in A_i^L} [d(r,s)]], \quad (k) \neq (r),(s),$$

where $\{(r,s)\}$ is a set of arcs which can possibly be replaced by arc (k,ℓ) without forming a loop among nodes $1, 2, \ldots, n_i$ but forms a loop with node (T_i).

4.2 If node (k) is out-of-kilter high, compute the increase in the cost of minimum 1-tree such that

$$\delta_i^L(k) = \min_{(k,\ell)\in A_i^L} [\min_{(r,s)\notin A_i^L} [d(r,s)] - d(k,\ell)], \quad (k) \neq (r),(s),$$

where $\{(r,s)\}$ is a set of arcs which can possibly replace arc (k,ℓ) without forming a loop among nodes $1,2, \ldots, n_i$ but forms a loop with node (T_i).

Step 5: Select a node having the minimum cost increment for the i-th group.

5.1 Find the minimum increment such that

$$\delta_i^*(k) = \min_{(k)} [\delta_i^L(k)].$$

If a tie exists select a node by any particular rule.

5.2 Let level L = L+1 and update the lower bound on the cost of tour such that

$$C_i^L = C_i^{L-1} + \delta_i^*(k).$$

Step 6: Update the distance matrix and associated minimum 1-tree(s) for the i-th group.

6.1 If the selected node (k) is out-of-kilter low, update the distance matrix such that

$$\hat{\delta}(k,\ell) = d(k,\ell) - \delta_i^*(k),\ (\ell) = 1,2,\ \ldots,\ n_i,\ T_i;\ (\ell) \neq (k),$$

and construct the corresponding minimum 1-tree(s). Then, go to step 7.

6.2 If the selected node (k) is out-of-kilter high, update the distance matrix such that

$$\hat{d}(k,\ell) = d(k,\ell) + \delta_i^*(k),\ (\ell) = 1,2,\ \ldots,\ n_i,\ T_i;\ (\ell) \neq (k),$$

and construct the corresponding minimum 1-tree.

Step 7: Check for minimum tour for the i-th group.

7.1 If all nodes in any minimum 1-tree are in-kilter, identify the tour and corresponding cost such that

$$C_i^* = C_i^L.$$

7.2 If minimum tours are obtained for all the groups of demand points go to step 8.

7.3 If no minimum 1-tree consists of all in-kilter nodes, go to step 3.

Step 8: Decompose the minimum tour into subtours for the i-th group.

8.1 If minimum tour of a group is a feasible tour, identify the corresponding route.

8.2 Compute the increments in the distance of minimum tour due to cut at particular arc such that

$$\Delta_i(k,\ell) = d(k, T_i) + d(\ell, T_i) - d(k,\ell).$$

8.2 Compute the increments in the distance of the minimum tour due to cut at particular arc such that

$$\Delta_i(k,\ell) = d(k, T_i) + d(\ell, T_i) - d(k,\ell).$$

8.3 Initiate counter K = 0.

Step 9: Check for feasible subtours for the i-th group.

9.1 Select an arc with the minimum value of increment at which the cut is made according to the following conditions: (1) feasible subtours are formed, and (2) demand points are evenly distributed in each subtour. If neither of the conditions is satisfied, the cut with the next minimum increment is selected and checked for the above two conditions. The procedure is repeated till a cut is found.

9.2 Compute the lower bound on the cost of all subtours of the group such that

$$C_i^* = C_i^* + \Delta_i(k,\ell).$$

Step 10. Identify the subtours for the i-th group.

10.1 If any feasible subtour is formed, identify the subtour and eliminate the nodes of the feasible tour from consideration.

10.2 Update the counter such that

$$K = K + N[\Delta_i(k,\ell)]$$

where $N[\Delta_i(k,\ell)]$ is the number of nodes forming a feasible subtour when the cut at (k,ℓ) is considered.

Step 11. Check for all subtours.

11.1 If $K = n_i$, identify all feasible subtours.

11.2 If $K < n_i$, go to step 9 treating non-feasible subtour as a tour and consider it for cuts.

Step 12. Check for all groups.

12.1 If $i = p$, the solution is obtained and identify all the subtours, respective loads, distances and the total distance of all subtours.

12.2 If $i < p$, let $i = i + 1$ and go to step 2.

The solution will result in a feasible set of subtours and the total distance of travel. The load and distance of individual subtour, can be found out as the route is known. It should be pointed out, however, that this solution may or may not be the optimal one. Another interesting point to note is the fact that this algorithm is general in the sense that it can solve both single- and multi-terminal problems. In the former, steps 1 and 12 must be suppressed.

5.2 Sample Problem.

The above algorithm can be demonstrated by considering a 10-city and 3 terminals delivery problem. The associated symmetrical distance matrix, demands at various cities and the number and capacity of available carriers, are shown in Table 5.1. The step by step procedure is summarized below.

First a minimum 1-tree is constructed by choosing minimum distance arcs in sequence subject to condition that the arc under consideration should be such that the nodes of the arc belong to only one of the T_i nodes. The minimum p-tree (which turns out to be 2-tree) of the sample problem is shown in Figure 5.1 and corresponding groups of demand points can be recognized as

Group 1: Terminal T_2: 2,6,8,9

Group 2: Terminal T_3: 1,3,4,5,7,10

The starting distance matrix for each group is shown in Tables 5.2 and 5.5.

The next step is to construct the minimum 1-tree for the first group of demand points as shown in Figure 5.2. At this point, the algorithm is initiated by setting level L = 0 for the minimum 1-tree. The lower bound on the distance of tour for the first group is computed at level 0 such that

$$C_i^0 = \sum_{(k,L)} d(k,\ell), \qquad (k,\ell) \in A_i^0 ,$$

where index i refers to an individual group and A_i^0 is the corresponding minimum 1-tree at level 0. Hence

Table 5.1 Distance Matrix and Demand for Three Terminals and 10 Demand Points Delivery Problem.

2	695									
3	331	502								
4	152	478	284							
5	172	483	167	121						
6	539	272	301	476	367					
7	217	569	125	216	114	394				
8	376	252	258	251	213	250	323			
9	364	290	211	276	192	200	279	70		
10	132	572	388	110	227	608	317	368	408	
Terminal 1 (T_1)	315	388	296	158	186	391	301	151	193	239
Terminal 2 (T_2)	409	336	172	354	234	130	264	170	95	509
Terminal 3 (T_3)	107	571	224	154	108	468	113	321	300	207
Nodes	1	2	3	4	5	6	7	8	9	10
Demands	12	5	8	15	6	7	14	4	13	8

Table Showing Available Number of Trucks and Corresponding Capacities.

Truck	40	30	20
Available	1	3	10

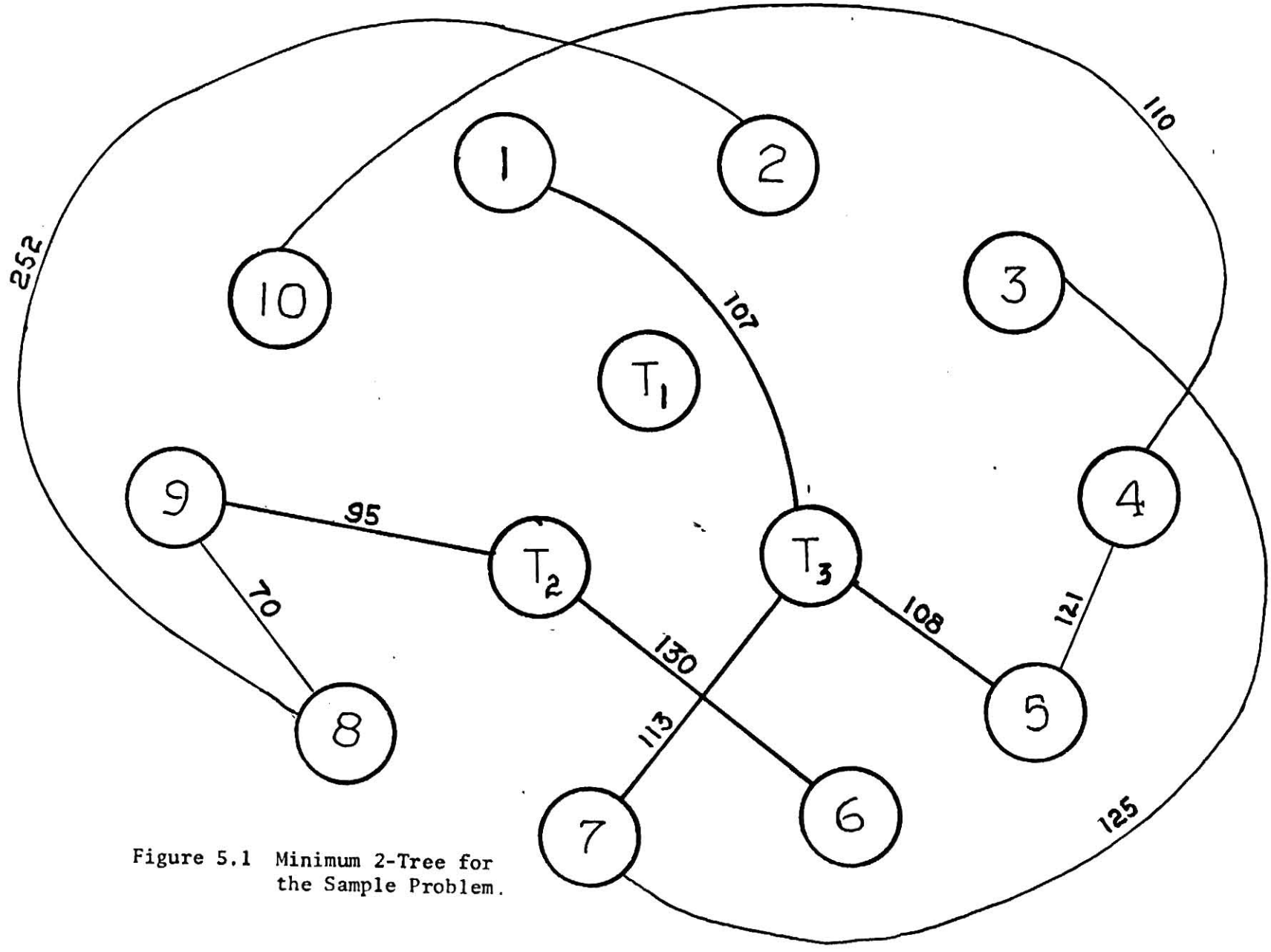


Figure 5.1 Minimum 2-Tree for the Sample Problem.

Table 5.2 Distance Matrix for the First Group with Terminal T_2.

6	272			
8	252	250		
9	290	200	70	
T2	336	130	170	95
Nodes ↳	2	6	8	9

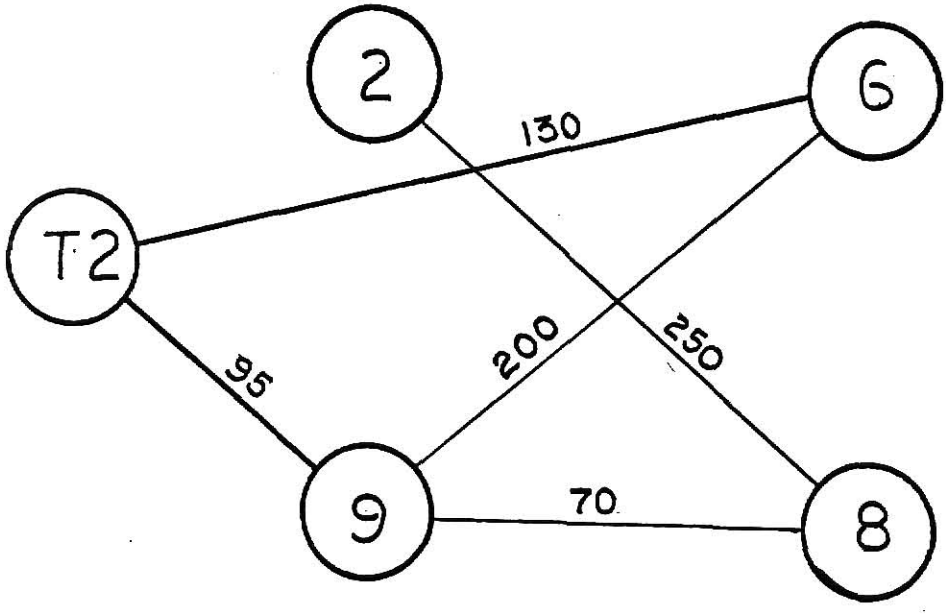


Figure 5.2. Initial Minimum 1-tree for the First Group with Terminal T_2.

$$C_1^0 = 252 + 200 + 70 + 130 + 95 = 747.$$

At level 0, it can be seen from the minimum 1-tree that nodes (2) and (9) are out-of-kilter low and out-of-kilter high, respectively. The cost increment for out-of-kilter nodes are computed as follows. For out-of-kilter low node (k), the increase in the cost of minimum 1-tree is computed such that

$$\delta_i^L(k) = \min_{(k,\ell)\notin A_i^L} [d(k,\ell) - \max_{(r,s)\in A_i^L} [d(r,s)]] \qquad (k) \neq (r),(s)$$

Hence for out-of-kilter low node (2),

$$\delta_1^0(2) = \min_{(2,\ell)\notin A_1^0} [d(2,\ell) - \max_{(r,s)\in A_1^0} \{d(r,s)\}]$$

$$= 72.$$

However, if node (k) is out-of-kilter high, the increase in the cost of the minimum 1-tree is computed such that

$$\delta_i^L(k) = \min_{(k,\ell)\in A_i^L} [\min_{(r,s)\notin A_i^L} d(r,s) - d(k,\ell)], \quad (k) \neq (r),(s)$$

Or the increment due to out-of-kilter high node (9) is computed such that

$$\delta_1^0(9) = \min_{(9,\ell)\in A_1^0} [\min_{(r,s)\notin A_1^0} [d(r,s)] - d(9,\ell)]$$

$$= 50.$$

At this point, a node having the minimum cost increment is selected for updating the distance matrix. Node (9) is selected as the node which

yields the minimum increment

$$\delta_1^*(k) = \min_{(k)} [\delta_1^0(k)], \quad k = 2,9$$

$$= \min_{(k)} [72,502 = \delta_1^*(9) = 50.$$

After finding $\delta_i^*(k)$, level L is increased by one and the lower bound on the cost of tour is updated as

$$C_i^L = C_k^{L-1} + \delta_i^*(k)$$

or

$$C_1^1 = C_1^0 + \Delta_1^*(k) = 747 + 50 = 797$$

If at any level, there is a tie for minimum $\delta_i^*(k)$, a node is selected by any particular rule. $\delta_i^*(k)$ is used to update the distance matrix such that

$$\hat{\delta}(k,\ell) = d(k,\ell) \pm \delta_i^*(k), \quad (\ell) = 1,2, \ldots, n_i, T_i \; j(\ell) \neq (k),$$

where $\delta_i^*(k)$ is used if node (k) is out-of-kilter high

$\delta_i^*(k)$ is used if node (k) is out-of-kilter low.

In our sample problem, the distance matrix for the first group of demand points will be updated for level 1 such that

$$\hat{d}(9,\ell) = d(9,\ell) + \delta_1^*(9), \quad (\ell) = 2,6,8,9$$

$$= d(9,\ell) + 50,$$

and consequently, a minimum 1-tree(s) is constructed from the updated matrix. The updated distance matrix and corresponding minimum 1-tree(s) for first group at level 1 are shown in Table 5.3 and Figure 5.3, respectively. At this point, it is checked if the minimum 1-tree is a

Table 5.3 Updated Distance Matrix for First Level for the First Group with Terminal T_2.

6	272			
8	252	250		
9	340	250	120	
T2	336	130	170	145
Nodes ↳	2	6	8	9

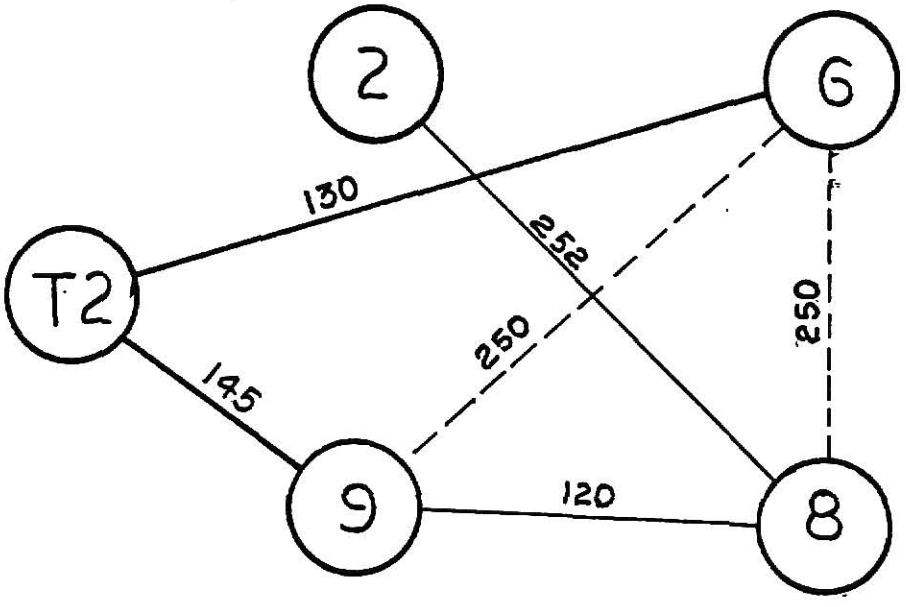


Figure 5.3 Updated Minimum 1-Tree for First Level for the First Group with Terminal T_2.

tour. If at the particular level all nodes in any minimum 1-tree are in-kilter, the tour is identified and the associated cost is $C_i^* = C_i^L$. However, if no minimum 1-tree forms a tour, the procedure is repeated till the level at which a minimum 1-tree becomes a tour. At level 1, minimum 1-trees of group 1 do not form tour as yet. But at level 2, the following tour is identified from the minimum 1-tree:

$$T_2 - 6 - 2 - 8 - 9 - T_2.$$

The summary of the computational steps of phase II for group 1 is shown in Table 5.4.

The next step is to decompose the minimum tour into a set of feasible subtours such that the contraints of capacity and number of carriers are satisfied. In our sample problem, as the sum of demands associated with nodes in group 1 is within the truck capacity, the minimum tour of this group is the feasible tour and no further consideration (or cuts) is necessary.

The same procedure is repeated for the next group of demand points. For group 2 of the sample problem, the initial distance matrix with terminal T_3 and initial minimum 1-tree are shown in Table 5.5 and Figure 5.4, respectively. The summary of iterations to obtain the minimum tour from the initial minimum 1-tree is summarized in Table 5.6. The minimum tour for this group is identified as

$$T_3 - 7 - 3 - 5 - 4 - 10 - 1 - T_3$$

with the total distance of 875. After the minimum tour is identified, it is considered for cuts. The associated increments are shown in Table 5.7. Arc (4,5) is chosen for cut as it has minimum increment

Table 5.4 Summary of Results for Minimum Tour of First Group.

Level L	Number of Minimum 1-Trees	Candidates Arcs	Arcs Comprising the Minimum 1-Tree $(k,\ell) \in A_1^L$	Out-of-Kilter Nodes: High Node (k)	High Cost Increment $\delta_1^L(k)$	Low Node (k)	Low Cost Increment $\delta_1^L(k)$	Minimum Cost Increment $\delta_1^*(k)$	Selected Node (k^*)	Lower Bound on the Distance of Tour C_1^L
0	1		$(T_2,6),(T_2,9),(2,8)$ $(6,9),(8,9)$	9	50	2	72	50	9	747
1	2	(6,9)	$(T_2,6),(T_2,9),(2,8)$ $(6,9),(8,9)$			2	22	22	2	797
		(6,8)	$(T_2,6),(T_2,9),(2,8)$ $(6,8),(8,9)$							
2	3	(6,9)	$(T_2,6),(T_2,9),(2,8)$ $(6,9),(8,9)$							819
		(6,8)	$(T_2,6),(T_2,9),(2,8)$ $(6,8),(8,9)$							
		(2,6)	T_2-6-2-8-9-T_2 **							

** Minimum 1-tree is a tour.

Table 5.5 Distance Matrix for the Second Group with Terminal T_3.

3	331					
4	152	284				
5	172	167	121			
7	217	125	216	114		
10	132	388	110	227	301	
T3	107	224	154	108	113	207
Nodes	1	3	4	5	7	10

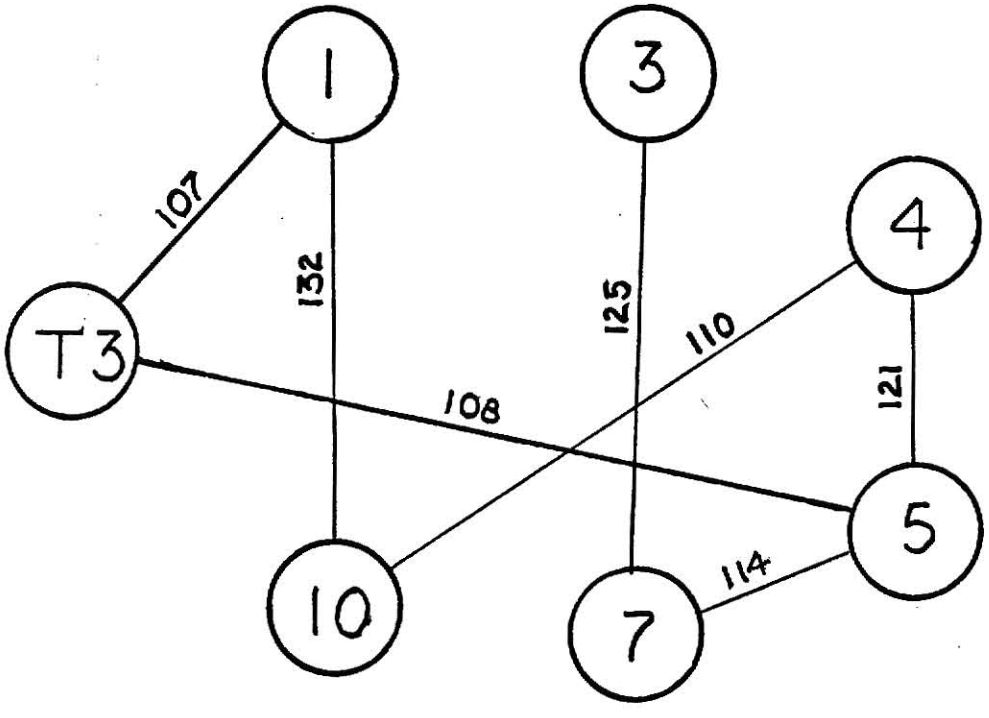


Figure 5.4 Initial Minimum 1-Tree for the Second Group with Terminal T_3.

Table 5.6 Summary of Results for Minimum Tour of Second Group.

Level	Number of Minimum 1-Trees	Candidate Arcs	Arcs Comprising the Minimum 1-Tree	Out-of-Kilter Nodes				Minimum Cost Increment	Selected Node	Lower Bound on the Distance of Tour
				High		Low				
				Node	Cost Increment	Node	Cost Increment			
L			$(k,\ell) \in A_2^L$	(k)	$\delta_2^L(k)$	(k)	$\delta_2^L(k)$	$\delta_2^*(k)$	(k^*)	C_2^L
0	1		(T3,1),(T3,5),(1,10) (3,7),(4,5),(4,10), (5,7)	5	5	3	53	5	5	817
1	2	(5,T3)	(T3,1),(T3,5),(1,10) (3,7),(4,5),(4,10), (5,7)			3	53	53	3	822
		(7,T3)	(T3,1),(T3,7),(1,10) (3,7),(4,5),(4,10), (5,7)			3	53	53	3	822
2	4	(5,T3) (5,7)	(T3,1),(T3,5),(1,10) (3,7),(4,5),(4,10), (5,7)							
		(5,T3) (3,5)	(T3,1),(T3,5),(1,10) (3,7),(4,5),(4,10), (3,5)							875
		(7,T3) (5,7)	(T3,1),(T3,7),(1,10) (3,7),(4,5),(4,10), (5,7)							
		(7,T3) (3,5)	T3-7-3-5-4-10-1-T3**							

** Minimum 1-tree is a tour.

and the cut at (4,5) results in two feasible subtours. The subtours are T_3 - 1 - 10 - 4 - T_3 and T_3 - 5 - 3 - 7 - T_3 with distances of 503 and 513, respectively. As the value of p is 2 (that is, there are only two groups) the final solution is reached and the feasible set of subtours, the corresponding loads and distances are dispalyed in Table 5.8. Table 5.9 shows the truck availability and the allocation.

5.3 Computational experiments:

A number of experiments were carried out to test the computational feasibility and the quality of solution produced by the proposed algorithm. The multi-terminal algorithm has been coded in FORTRAN IV for the IBM 360/50 computer. The computer program is general in the sense that it can solve either multi- or single-terminal problems. The 36 problems tried, ranged in size from 1 to 5 terminals and 6 to 25 demand points. Table 5.10 displays the solutions of the problems. The performance of the algorithm is evaluated with respect to the quality of solution by comparing the solutions obtained with known solutions.

5.4 Comparative Evaluation.

It would be appropriate at this point, to offer a brief discussion about the existing techniques with which the proposed algorithm is going to be compared. In the heuristic approach which is based on "savings of distance" and initiated by Clarke and Wright [4], the saving in the distance resulting due to connecting two demand points is computed for each combination of demand points. Two demand points having maximum savings are linked if the linking is feasible with respect to other constraints. Tillman and his associates [5,12,16,17] developed

Table 5.7 Possible Increments Resulting Due to Cut at a Particular Arc.

No.	Arc for the Cut (k,ℓ)	Increase Due to Cut $\Delta_i(k,\ell)$	Bound on all Subtours
1	(1,10)	182	
2	(10,4)	251	
3	(4,5)	141*	875
4	(5,3)	275	
5	(3,7)	212	

*arc chosen for cut

Table 5.8. Table Showing Final Solution to the Sample Delivery Problem.

No.	Terminal	Route	Load	Distance
1	2	T_2 - 6 - 2 - 8 - 9 - T_2	29	819
2	3	T_3 - 1 - 10 - 4 - T_3	35	503
3	3	T_3 - 5 - 3 - 7 - T_3	28	513

Table 5.9 Capacity and Availability of Trucks at End of First Cut.

Truck	40	30	20
Available	1	3	10
Allocated	1	2	0

Table 5.10 Summary of Distances for all Delivery Problems.

Problem No.	Number of Demand Points	Best Solution		Computational Time (min)
		Saving Method	Proposed Method	
1	6	*	1016	0.92
2	6	394	394	1.02
3	6	*	337	0.91
4	6	*	306	0.55
5	6	*	1808	2.77
6	6	*	974	0.78
7	6	*	2260	1.02
8	7	*	1193	0.97
9	8	*	2228	0.90
10	9	*	918	8.25
11	9	*	1808	2.84
12	10	76	76	10.62
13	10	2470	2470	9.6
14	10	2156	2156	4.21
15	10	1679	1679	5.45
16	10	4109	3941	1.40
17	10	*	2626	12.25
18	10	*	2550	7.9
19	10	*	3102	12.5
20	10	*	691	12.2
21	14	*	1141	15.6
22	25	9604	9553	30.72
23	25	8325	8582	61.44
24	25	2289	2295	25.56
25	25	2638	2638	36.12
26	25	8901	8901	25.66
3 Terminals and 10 Demand Points Problems.				
27	10	59	59	1.68
28	10	1835	1835	1.42
29	10	2042	1857	1.34
30	10	1706	1638	1.38
31	10	3490	3468	1.52
5 Terminals and 25 Demand Points Problems.				
32	25	6832	6857	25.35
33	25	6099	6695	21.58
34	25	1377	1470	13.71
35	25	1722	2238	25.85
36	25	6653	6905	10.49

*Solutions by savings approach are not known.

modified versions of the savings approach with more than one decision and different alternatives. The alternatives are selected from the saving matrices such that alternative 1 is the pair of demand points with the maximum feasible savings, alternative 2 is the pair of demand points with the second highest feasible saving and so forth. The next step is the construction of a decision tree for each alternative by considering the links with the next maximum feasible savings and the next maximum feasible saving depending on whether it is 2-decision or 3-decision look ahead criterion. The savings for each branch of each alternative are added and the particular branch with maximum sum of savings is chosen.

Table 5.11 displays the distances for the single-terminal problems by different techniques. Only those single-terminal problems are compared of which solutions by other techniques are known. It can be easily observed that except for problems 23 and 24, the solutions by the proposed approach are equal or superior to those obtained by the savings approach. The efficiency of the algorithm is determined for each solution as ratio of the best solution by savings approach to that of the proposed algorithm. The efficiency of most of the solutions is equal to or more than 100%.

The solutions of the second set of experiments with 3 terminals and 10 demand points are summarized in Table 5.12. A similar observation can be made regarding the solutions by the proposed approach. The average efficiency is found to be 103.5%.

The solutions of 5 terminals and 25 demand points delivery problem are displayed in Table 5.13. The solutions by the proposed algorithm

Table 5.11 Summary of Distances for the Single-Terminal Delivery Problems.

Problem No.	Savings Approach [4]	Modified Savings with 2-Decision Look-Ahead Approach [5,17]			Modified Savings with 3-Decision Look-Ahead Approach [12]			Proposed Approach	Efficiency††
		2 Alternatives	3 Alternatives	5 or 10 Alternatives	2 Alternatives	3 Alternatives	5 or 10 Alternatives		
12	76	76	76	76†	76	76	76	76	100
13	2846	2470	2470	2470†	2470	2470	2470	2470	100
14	2156	2156	2156	2156†	2156	2156	2156	2156	100
15	1813	1679	1679	1679†	1679	1679	1679	1679	100
16	4146	4146	4121	4121†	4146	4109	4109	3941	104.5
22	9604	9990	9848	9848*	9709	9689	9689	9553	101.9
23	8383	8325	8325	8325*	8325	8325	8325	8582	97.4
24	2289	2289	2289	2289*	2289	2289	2289	2295	99.8
25	2788	2788	2788	2788*	2657	2638	2638	2638	100
26	8901	8901	8901	8901*	8901	8901	8901	8901	100

†Problems 12 - 16 are solved by using 5 alternatives.

*Problems 22 - 26 are solved by using 10 alternatives.

††Efficiency of the solution is defined as the quotient of the best solution by savings approach to that by proposed method.

Table 5.12 Summary of Distances for the 3 Terminal and 10 Demand Point Problems

Problem No.	Tillman's method	Herring's 3-Decision Look-Ahead Method			Proposed Algorithm	
		2 alternatives	3 alternatives	5 alternatives	Distance	Efficiency*
27	61	61	59	59	59	100
28	1835	1835	1835	1835	1835	100
29	2045	2045	2087	2087	1857	110
30	1715	1706	1706	1706	1638	104
31	3629	3490	3490	3490	3468	103.5

Average Efficiency = 103.5

Table 5.13 Summary of Distances for the 5 Terminal and 25 Demand Point Problems

Problem No.	Tillman's method	Herring's 3-Decision Look-Ahead Method				Proposed Algorithm	
		2 alternatives	3 alternatives	5 alternatives	10 alternatives	Distance	Efficiency*
32	7005	6832	6832	6832	6832	6857	99.9
33	6182	6099	6099	6099	6099	6695	91.0
34	1377	1377	1377	1377	1377	1470	34.0
35	1824	1722	1824	1828	1828	2236	77.5
36	6840	6783	6783	6783	6653	6915	96.75

Average Efficiency = 91.03

*Efficiency of the solution is defined as the quotient of solution by savings approach to that by proposed method.

are not as good as those by the savings approach. The average efficiency is found out as 91.03%. The probable cause for this inconsistency in the quality of solution could be the particular nature of the data for this set of problems. As a result no conclusive comment can be made as the sample for comparison is very small.

5.5 Conclusions:

To the author's knowledge, this is the first time, the basic concepts of the graph-theoretical approach is applied to the delivery problem. Thus applied problems such as the single- and multi-terminal delivery problems are brought within the highly developed mathematical discipline of graph theory.

Although the results obtained for large problems are somewhat inferior to those obtained by the savings approach, there is no doubt that the imposition of suitable constraints in addition to distances, on the minimum 1-tree would improve thie quality of solution.

REFERENCES

1. Balinski, M. L., and Quandt, R. E., "On an Integer Program for a Delivery Problem," Operations Research, Vol. 12, No. 2, 1964, pp. 300-304.

2. Barachet, L. L., "Graphical Solution of the Travelling-Salesman Problem," Operations Research, Vol. 5, No. 6, 1957, pp. 841-845.

3. Bellman, R., "On a Routing Problem," Quarterly Journal of Applied Mathematics, Vol. 16, No. 1, 1958, pp. 87-90.

4. Clarke, G., and Wright, J. W., "Scheduling of Vehicles from a Central Depot to a Number of Delivery Points," Operations Research, Vol. 12, No. 4, 1964, pp. 568-581.

5. Cochran, H., "Optimization of a Carrier Routing Problem," Master's Thesis, Kansas State University, Manhattan, Kansas, 1967.

6. Dantzig, G. B., and Ramser, H. H., "The Truck Dispatching Problem," Management Science, Vol. 6, No. 1, 1959, pp. 80-91.

7. Gomory, R. E., "All-Integer Programming Algorithm," RC-189, IBM Research Center, N.Y. City, N.Y., January 29, 1960.

8. Gomory, R. E., "An Algorithm for Integer Solutions to LInear Programs", Recent Advances in Mathematical Programming, (edited by R. L. Graves and P. Wolfe), McGraw-Hill Book Company, Inc., New York, 1963.

9. Gonzales, R. H., "Solution of the Traveling Salesman Problem by Dynamic Programming on the Hypercube," Interim Tech. Report No. 18, Operations Research Center, M.I.T., Cambridge, Mass., 1962.

10. Held, M., and Karp, R. M., "A Dynamic Programming Approach to Sequencing Problems," J. Soc. ind. appl. Math,Vol. 10, 1962, pp. 196-210.

11. Held, M., and Karp, R. M., "The Traveling-Salesman Problem and Minimum Spanning Trees," Operations Research, Vol. 18, No. 6, 1970, pp. 1138-1162.

12. Herring, R., "Evaluation of Some Heuristic Look Ahead Rules for Multiple Terminal Delivery Problems," Master's Thesis, Kansas State University, Manhattan, Kansas, 1970.

13. Little, J. D. C., Murty, K. G., Sweeney, D. W., and Karel, C., "An Algorithm for the Traveling Salesman Problem," Operations Research, Vol. 11, No. 6, 1963, pp. 972-989.

14. Miller, C. E., Tucker, A. W., and Zemlin, A. W., "Integer Programming Formulation of Traveling Salesman Problems," Association for Computing Machinery, Vol. 7, No. 4, 1960, pp. 326-329.

15. Pierce, J. F., "Direct Search Algorithms for Truck - Dispatching Problems," Transportation Research, Vol. 3 (1969), Transportation Research, Vol. 3, No. 1, 1969, pp. 1-42.

16. Tillman, F. A., "The Multiple Terminal Delivery Problem with Probabilistic Demands," Transportation Science, Vol. 3, 1969, pp. 192-204.

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

18. Tillman, F. A., "Dynamic Programming Solution to the School Bus Scheduling Problem," Unpublished Working Paper, Kansas State University, Manhattan, Kansas, 1965.

APPENDIX A

COMPUTER PROGRAM

COMPUTER PROGRAM

A general multi-terminal delivery problem algorithm, as developed in Chapter III, is coded in FORTRAN IV for 360/50 computer. The same program is used to solve single-terminal problem by specifying the number of terminals equal to one. The main program includes the following external subroutines:

IGRUP. This subroutine is called to form minimum p-tree and assign the demand points to terminals.

IARANG. This subroutine arranges the demand points grouped by subroutine IGRUP for easy identification.

ITREE. This subroutine is called to construct minimum 1-tree(s).

KILTER. This subroutine identifies the out-of-kilter nodes.

JGAPLO. This subroutine computes the minimum cost increments for out-of-kilter low nodes.

JGAPHI. This subroutine is called to compute minimum cost increments for out-of-kilter high nodes.

IUPDAT. The subroutine IUPDAT updates the distance matrix by minimum of increments computed by subroutines JGAPLO and JGAPHI.

IROUT. This subroutine is called to form feasible routes by cuts at the arcs of minimum tour.

IFIRST. This subroutine lasts if the cut at the arc with the minimum increment forms feasible routes.

ISECND. This subroutine is called to test if the cut with next minimum increment forms feasible routes.

IHALF. This subroutine is called to form non-feasible subtours to be considered for further cuts.

IASCND. This subroutine orders the increments due to cut at particular arcs of minimum tour.

INPUT INSTRUCTIONS. The necessary data is to be supplied in following order.

Card 1. Number of problems to be run.

Card 2. Number of demand points, number of terminals and number of types of trucks.

Card 3. Demands at various points starting from demand point 1.

Card 4. Truck capacities with highest truck capacity first, next highest second etc.

Card 5. Number of trucks of each capacity and in same order as above [card 4].

Card 6. Distance matrix. Only half distance matrix is necessary as it is a symmetrical, last row(s) in the distance matrix is distance(s) from terminal(s).

All data is supplied with the same format [16I5].

Output.

The output has been condensed as much as possible so that a minimum of output is obtained. Intermediate results can be easily obtained by inserting write statements to write out the desired information. The output obtained with the program shown below consists of the following.

[1]. Echo check: Supplied data is printed out. That is, demands at various demand points, types of trucks, truck capacities and number of trucks of each capacity and the distance matrix.

[2]. Routes determined by algorithm.

[3]. The final truck allocation table for all routes.

Appendix B lists output of all single-terminal delivery problems [26 problem]. The data and routes determined by algorithmmfor multi-terminal delivery problems are displayed in Appendix C.

It should be noted that, the way program is coded it is necessary to have at least two types of trucks. If only one type of truck [only one capacity] is available make number of types of trucks equal to two and let the second type of truck be of zero capacity.

```
C*************************************************************
C
C     PROGRAM FOR SOLVING CARRIER ROUTING PROBLEM BY
C     GRAPH-THEORETIC APPROACH
C
C*************************************************************
C
C     PROGRAMMED BY
C     P. BHATT
C     DEPARTMENT OF INDUSTRIAL ENGINEERING
C     KANSAS STATE UNIVERSITY
C
C*************************************************************
C     VARIABLES EXPLAINATION
C*************************************************************
C        **** INPUT VARIABLES ****
C     NPROB.........NUMBER OF PROBLEMS TO BE RUN
C     N.............NUMBER OF DEMAND POINTS
C     ITER..........NUMBER OF TERMINALS
C     ITYPE.........NUMBER OF TYPES OF TRUCKS AVAILABLE
C     ICAP(I).......TRUCK CAPACITY OF TYPE I
C     NTRCK(I)......NUMBER OF TRUCKS OF TYPE I
C     IDEMND(I).....DEMAND AT POINT I
C     IDIST(I,J)....DISTANCE BETWEEN POINTS I AND J
C                   ( J > N REPRESENTS TERMINAL(S) ).
C*************************************************************
C        **** PROGRAM VARIABLES ****
C     LINKS(I,J)....STARTING NODE OF ARC J OF TREE I
C     LINKF(I,J)....END NODE OF ARC J OF TREE I
C     KOK(I,J)......DEGREE OF NODE J OF TREE I
C     KD(I,J).......STATE OF NODE J OF TREE I
C                   IF KD < 0 NODE IS OUT-OF-KILTER LOW
C                   IF KD > 0 NODE IS OUT-OF-KILTER HIGH
C     IRT(I,J)......NON-FEASIBLE TOUR ELEMENT J OF TOUR I
C     IRT(I,J)......ELEMENT J OF NON-FEASIBLE SUBTOUR I
C     IROUT(I,J)....ELEMENT J OF ROUTE I
C     NCOMP(I)......NUMBER OF DEMAND POINTS IN ROUTE I
C     JS(I,J).......ELEMENT J OF GROUP I OF MINIMUM P-TREE
C*************************************************************
C        **** INPUT INSTRUCTIONS ****
C     CARD 1       NPROB                          FORMAT(16I5)
C     CARD 2       N,ITER,ITYPE                   FORMAT(16I5)
C     CARD 3       (IDEMND(I),I=1,N)              FORMAT(16I5)
C     CARD 4       (ICAP(I),I=1,ITYPE)            FORMA (16I5)
C     CARD 5       NTRCK(I),I=1,ITYPE)            FORMAT(16I5)
C     CARD 6       (IDIST(I,J),I=1,J-1)           FORMAT(16I5)
C     NOTE: ONLY HALF DISTANCE MATRIX IS TO BE USED.
C           J > N REPRESENTS TERMINAL(S).
C     REPEAT FROM CARD 2 FOR EACH PROBLEM
C*************************************************************
      COMMON IDIST(25,30),LDIST(25,30),KPR(25,30),IDEMND(25),
     *LDEMND(25),ICAP(9),NTRCK(9)
      COMMON ITER,ITYPE,N
      COMMON LINKS(50,26),LINKF(50,26),KD(50,25),KOK(50,25)
      COMMON JS(5,15),KOUNT(5),LD(25),LS(450),LF(450),B(25),
```

```
     *MF(200),MS(200),LEAD(25),IB(200),MR(25)
      COMMON IROUT(9,5),NCOMP(6),IIN(20,25),IRT(20,26),KL(26),
     *INCR(20,26),IRMIN(20,25),KLINK(26)
      COMMON JJGAP(50,25),LINKSN(26),LINKFN(26),L(26)
      DIMENSION ITRCK(9)
      INTEGER FORM(15),DIGIT(25)
C
C
C   ***FORMAT CARDS
C
C
      DATA FORM/'(/,9','X,I1',',5X,',',I1,7','X,1H','T,I1',',   ',','    '
     *,'I4,3','H  T',',I1,',','    ','X,I3',',2X,',','I5) '/
      DATA DIGIT/'   1',','  2',','  3',','  4',','  5',','  6',','  7',','  8'
     *,'  9',',  10',',  11',',  12',',  13',',  14',',  15',',  16',',  17',
     *'  18',',  19',',  20',',  21',',  22',',  23',',  24',',  25'/
    10 FORMAT(//,8X,25HNUMBER OF DEMAND POINTS =,I2,5X,21HNUMBER OF TERMI
     *NALS =,I2)
     5 FORMAT('1',////,25X,7HPROBLEM,I3)
    15 FORMAT(//,8X,6HDEMAND,3X,6HDEMAND,3X,8HDISTANCE,2X,8HDISTANCE,2X,8
     *HDISTANCE,2X,8HDISTANCE,2X,8HDISTANCE)
    20 FORMAT(8X,5HPOINT,4X,6HAT THE,4X,6HTO THE,4X,6HTO THE,4X,6HTO THE,
     *4X,6HTO THE,4X,6HTO THE)
    25 FORMAT(18X,5HPOINT,3X,48HTERMINAL  TERMINAL  TERMINAL  TERMINAL  T
     *ERMINAL)
    26 FORMAT(27X,5HNO. (,I1,9H)   NO. (,I1,9H)   NO. (,I1,9H)   NO. (,I1
     *,9H)   NO. (,I1,1H))
    30 FORMAT(7X,67H-------  --------  --------  --------  --------  ----
     *----  --------)
    35 FORMAT(////,13X,15HDISTANCE MATRIX,///)
    40 FORMAT(16I5)
    45 FORMAT(/////,13X,6HTRUCKS)
    50 FORMAT(/,6X,17HCAPACITY   NUMBER)
    55 FORMAT(6X,17H--------   ------)
   110 FORMAT(9X,I2,8X,I2,1X,5I10)
   120 FORMAT(8X,I3,8X,I2)
   130 FORMAT(6X,1H(,I2,1H),24I5)
   135 FORMAT('1',//////////,25X,14HFINAL SOLUTION)
   137 FORMAT(25X,14H----- --------,/)
   140 FORMAT(/,6X,58(1H*))
   141 FORMAT(6X,58(1H-))
   143 FORMAT(/,8X,55HNO. TERMINAL          ROUTES                LOAD DIST
     *ANCE)
   144 FORMAT(/,39X,16HTOTAL DISTANCE =,I5)
   145 FORMAT(/////,8X,39HTRUCK AVAILABILITY AND ALLOCATION TABLE)
   147 FORMAT(8X,39H----- ----------- --- ---------- -----,/)
   148 FORMAT(/,7X,74(1H*))
   172 FORMAT(/,8X,5HTRUCK,4X,9I7)
   173 FORMAT(/,8X,9HAVAILABLE,9I7)
   174 FORMAT(/,8X,9HALLOCATED,9I7)
C
C      READ THE NUMBER OF PROBLEMS
C
       READ(5,40)NPROB
       DO 1000 NPR=1,NPROB
```

```
C
C      READ THE SIZE OF PROBLEM
C
       READ(5,40)N,ITER,ITYPE
C
C      READ THE DEMANDS AT DEMAND POINTS
C
       READ(5,40)(IDEMND(I),I=1,N)
       DO 48 I=1,N
   48  LDEMND(I)=IDEMND(I)
       LTR=N
C
C      READ THE TRUCK CAPACITIES
C
       READ(5,40)(ICAP(I),I=1,ITYPE)
C
C      READ THE NUMBER OF DIFFERENT CAPACITY TRUCKS
C
       READ(5,40)(NTRCK(I),I=1,ITYPE)
C
C      READ THE DISTANCES
C
       N2=N-1
       DO 51 J=2,N
       J1=J-1
       READ(5,40)(IDIST(I,J),I=1,J1)
       DO 49 I=1,J1
       KPR(I,J)=IDIST(I,J)
   49  LDIST(I,J)=IDIST(I,J)
   51  CONTINUE
       DO 65 JJ1=1,ITER
       J=JJ1+N
       READ(5,40)(IDIST(I,J),I=1,N)
       DO 64 I=1,N
       KPR(I,J)=IDIST(I,J)
   64  LDIST(I,J)=IDIST(I,J)
   65  CONTINUE
       WRITE(6,5)NPR
       WRITE(6,10)N,ITER
       WRITE(6,15)
       WRITE(6,20)
       WRITE(6,25)
       WRITE(6,26)(I,I=1,ITER)
       WRITE(6,30)
       DO 105 I=1,N
  105  WRITE(6,110)I,IDEMND(I),(IDIST(I,(N+J)),J=1,ITER)
       WRITE(6,45)
       WRITE(6,50)
       WRITE(6,55)
       DO 112 I=1,ITYPE
       ITRCK(I)=NTRCK(I)
       WRITE(6,120)ICAP(I),NTRCK(I)
  112  CONTINUE
       WRITE(6,35)
       DO 115 J=2,N
```

```
      J1=J-1
      WRITE(6,130)J,(IDIST(I,J),I=1,J1)
  115 CONTINUE
      WRITE(6,135)
      WRITE(6,137)
      WRITE(6,140)
      WRITE(6,143)
      WRITE(6,140)
      ITCTL=0
      DO 66 I=1,N
   66 LD(I)=I
      IF(ITER.EQ.1)GO TO 69
C
C     SUBROUTINE IGRUP IS CALLED TO FORM MINIMUM P-TREE
C
      CALL IGRUP
   69 DO 70 ITERM=1,ITER
      IF(ITER.EQ.1)GO TO 59
      N=KOUNT(ITERM)
      IF(N.EQ.0)GO TO 70
C
C     SUBROUTINE IARNG IS CALLED TO ARRANGE THE DEMAND POINTS
C     IN EACH GROUP FOR EASY IDENTIFICATION
C
      CALL IARNG(ITERM)
      N2=N-1
      N1=N+1
      LD(N1)=LTR+ITERM
      DO 75 J=2,N1
      J1=J-1
      DO 72 I=1,J1
      IDIST(I,J)=LDIST(LD(I),LD(J))
      KPR(I,J)=IDIST(I,J)
   72 CONTINUE
   75 CONTINUE
      DO 76 I=1,N
      IDEMND(I)=LDEMND(LD(I))
   76 CONTINUE
   59 NTK=(N2)*(N2+1)/2
      IF(N.LE.2)GO TO 330
C
C     TO OBTAIN A MIN. ROUTE
C
C
C     THIS DO-LOOP IS FOR NO. OF ITERATIONS
C
      DO 60 IP=1,100
C
C     THE SUBROUTINE ITREE IS CALLED TO FORM A MINIMUM 1-TREE
C
      CALL ITREE(NTREE,NTK)
C
C     THE CHECK IS PERFORMED TO SEE WHETHER THE MINIMUM TOUR
C     IS OBTAINED
C
```

```
      DO 136 JC=1,NTREE
      DO 146 JD=1,N
      IF(JD.EQ.1)GO TO 146
      IF(KOK(JC,(JD-1)).NE.KOK(JC,JD))GO TO 136
      IF(JD.EQ.N)GO TO 500
      GO TO 146
  500 JE=JC
      GO TO 337
  146 CONTINUE
  136 CONTINUE
C
C     THE SUBROUTINE KILTER IS CALLED TO FIND OUT-OF-KILTER
C     NODES
C
      CALL KILTER(NTREE)
C
C     THE SUBROUTINE IGAPLO IS CALLED TO COMPUTE THE MINIMUM
C     INCREASE IN MINIMUM 1-TREE WHEN THE NODE IS OUT-OF-KILTER LOW
C
      CALL IGAPLO(NTREE,JGAPLO,JNODLO)
C
C     THE SUBROUTINE IGAPHI IS CALLED TO COMPUTE THE MINIMUM
C     INCREASE IN MINIMUM 1-TREE WHEN THE NODE IS OUT-OF-KILTER HIGH
C
      CALL IGAPHI(NTREE,JGAPHI,JNODHI)
      IF(JGAPLO.LT.JGAPHI)GO TO 310
      JNODE=JNODHI
      JGAP=JGAPHI
      GO TO 321
  310 JNODE=JNODLO
      JGAP=-JGAPLO
C
C     THE SUBROUTINE IUPDAT IS CALLED TO UPDATE THE DISTANCE MATRIX
C
  321 CALL IUPDAT(JNODE,JGAP)
      IF(IP.EQ.19)GO TO 170
   60 CONTINUE
  330 LOAD=0
      NM=1
      ID=IDIST(1,N1)
      DO 335 I=1,N
      IROUT(NM,I)=I
      LOAD=LOAD+IDEMND(I)
  335 CONTINUE
      IF(N.EQ.1)GO TO 336
      ID=ID+IDIST(1,2)
  336 ID=ID+IDIST(N,N1)
      NFINAL=N
      FORM(8)=DIGIT(NFINAL)
      K=5+(4-NFINAL)*4
      FORM(12)=DIGIT(K)
      WRITE(6,FORM)NM,ITERM,ITERM,(LD(IROUT(I,J)),J=1,NFINAL),ITERM,LOAD
     *,ID
      ITOTL=ITOTL+ID
      GO TO 70
```

```
  337 DO 162 J=2,N1
      J1=J-1
      DO 161 I=1,J1
      IDIST(I,J)=KPR(I,J)
      IDIST(J,I)=IDIST(I,J)
  161 CONTINUE
  162 CONTINUE
C
C     THE SUBROUTINE IROUTS IS CALLED TO FORM FEASIBLE ROUTS
C     BY APPLYING ' CUT ' AT THE ARC MINIMUM INCREASE
C
      CALL IROUTS(JE,NM)
      IF(NM.EQ.1)GO TO 164
      NM1=NM-1
      GO TO 175
  164 NM1=1
  175 DO 95 I=1,NM1
      NFINAL=NCOMP(I)
      ID=IDIST(IROUT(I,1),(N+1))
      J=1
      IF(NFINAL.EQ.1)GO TO 13
      DO 12 J=2,NFINAL
      ID=ID+IDIST(IROUT(I,(J-1)),IROUT(I,J))
   12 CONTINUE
      J=NFINAL
   13 ID=ID+IDIST(IROUT(I,J),(N+1))
      LOAD=0
      DO 14 J=1,NFINAL
      LOAD=LOAD+IDEMND(IROUT(I,J))
   14 CONTINUE
      FORM(8)=DIGIT(NFINAL)
      K=5+(4-NFINAL)*4
      FORM(12)=DIGIT(K)
      WRITE(6,FORM)I,ITERM,ITERM,(LD(IROUT(I,J)),J=1,NFINAL),ITERM,LOAD
     *,ID
      ITOTL=ITOTL+ID
   95 CONTINUE
      WRITE(6,141)
   70 CONTINUE
      WRITE(6,140)
      WRITE(6,144)ITOTL
      DO 1001 I=1,ITYPE
 1001 L(I)=ITRCK(I)-NTRCK(I)
      WRITE(6,145)
      WRITE(6,147)
      WRITE(6,148)
      WRITE(6,172)(ICAP(I),I=1,ITYPE)
      WRITE(6,148)
      WRITE(6,173)(ITRCK(I),I=1,ITYPE)
      WRITE(6,174)(L(I),I=1,ITYPE)
      WRITE(6,148)
 1000 CONTINUE
  170 STOP
      END
```

```
      SUBROUTINE IGRUP
C*************************************************************
C     THIS SUBROUTINE GROUPES DEMAND POINTS FOR EACH TERMINAL
C*************************************************************
      COMMON IDIST(25,30),LDIST(25,30),KPR(25,30),IDEMND(25),
     *LDEMND(25),ICAP(9),NTRCK(9)
      COMMON ITER,ITYPE,N
      COMMON LINKS(50,26),LINKF(50,26),KD(50,25),KOK(50,25)
      COMMON JS(5,15),KOUNT(5),LD(25),LS(450),LF(450),B(25),
     *MF(200),MS(200),LEAD(25),IB(200),MR(25)
      COMMON IRCLT(9,5),NCOMP(6),IIN(20,25),IRT(20,26),KL(26),
     *INCR(20,26),IRMIN(20,25),KLINK(26)
      COMMON JJCAP(50,25),LINKSN(26),LINKFN(26),L(26)
      N2=N-1
      NTK=(N2)*(N2+1)/2+N*ITER
      NS=N+1
      NF=N+ITER
      DO 99 I=1,ITER
      KOUNT(I)=0
   99 CONTINUE
C
C     TO ARRANGE ALL DISTANCES IN ASCENDING ORDER
C
      DO 210 IP=1,NTK
      IL=IP-1
      ITEMP=9999
      DO 110 J=2,N
      J1=J-1
      DO 130 I=1,J1
      IF(IL.LT.1)GO TO 119
      DO 120 JK=1,IL
      IF(LS(JK).EQ.I.AND.LF(JK).EQ.J)GO TO 130
  120 CONTINUE
  119 IF(IDIST(I,J).LT.ITEMP)GO TO 140
      GO TO 130
  140 II=I
      JJ=J
      ITEMP=IDIST(I,J)
  130 CONTINUE
  110 CONTINUE
      LS(IP)=II
      LF(IP)=JJ
      DO 150 J=NS,NF
      DO 160 I=1,N
      IF(IL.LT.1)GO TO 169
      DO 170 JK=1,IL
      IF(LS(JK).EQ.I.AND.LF(JK).EQ.J)GO TO 160
  170 CONTINUE
  169 IF(IDIST(I,J).LT.ITEMP)GO TO 180
      GO TO 160
  180 LS(IP)=I
      LF(IP)=J
      ITEMP=IDIST(I,J)
  160 CONTINUE
  150 CONTINUE
```

```
C
C     TO GROUPS THE VARIOUS DEMAND POINTS WITH VARIOUS TERMINALS
C
      NCHK=LF(IP)-N
      DO 162 I=1,ITER
      I1=KOUNT(I)
      IF(I1.LE.0)GO TO 162
      DO 161 J=1,I1
      IF(LS(IP).EQ.JS(I,J))GO TO 105
      IF(LF(IP).EQ.JS(I,J))GO TO 101
  161 CONTINUE
      GO TO 162
  105 DO 79 II=1,ITER
      ITST=N+1
      I2=KOUNT(II)
      IF(I2.EQ.0)GO TO 79
      DO 78 JJ=1,I2
      IF(LF(IP).EQ.JS(II,JJ).OR.LF(IP).GE.ITST)GO TO 210
   78 CONTINUE
   79 CONTINUE
      KOUNT(I)=KOUNT(I)+1
      JS(I,KOUNT(I))=LF(IP)
      NCHK=I
      GO TO 505
  101 DO 83 II=1,ITER
      I2=KOUNT(II)
      IF(I2.EQ.0)GO TO 83
      DO 81 JJ=1,I2
      IF(LS(IP).EQ.JS(II,JJ))GO TO 210
   81 CONTINUE
   83 CONTINUE
      KOUNT(I)=KOUNT(I)+1
      JS(I,KOUNT(I))=LS(IP)
      NCHK=I
      GO TO 505
  162 CONTINUE
      IF(LF(IP).LE.N)GO TO 210
      KOUNT(NCHK)=KOUNT(NCHK)+1
      JS(NCHK,KOUNT(NCHK))=LS(IP)
  505 ITST=N+1
      IP1=IP-1
      IF(IP1.EQ.0)GO TO 210
  159 DO 164 I=1,IP1
      KP=KOUNT(NCHK)
      DO 163 J=1,KP
      IF(LS(I).EQ.JS(NCHK,J))GO TO 165
      IF(LF(I).EQ.JS(NCHK,J))GO TO 166
  163 CONTINUE
      GO TO 164
  165 DO 269 II=1,ITER
      I2=KOUNT(II)
      IF(I2.EQ.0)GO TO 269
      DO 268 JJ=1,I2
      IF(LF(I).EQ.JS(II,JJ).OR.LF(I).GE.ITST)GO TO 164
  268 CONTINUE
```

```
269 CONTINUE
    KOUNT(NCHK)=KOUNT(NCHK)+1
    JS(NCHK,KOUNT(NCHK))=LF(I)
    GO TO 168
166 DO 279 II=1,ITER
    I2=KOUNT(II)
    IF(I2.EQ.0)GO TO 279
    DO 278 JJ=1,I2
    IF(LS(I).EQ.JS(II,JJ))GO TO 164
278 CONTINUE
279 CONTINUE
    KOUNT(NCHK)=KOUNT(NCHK)+1
    JS(NCHK,KOUNT(NCHK))=LS(I)
    GO TO 168
164 CONTINUE
    GO TO 200
168 GO TO 159
200 KK=0
    DO 202 LM=1,ITER
202 KK=KK+KOUNT(LM)
    IF(KK.EQ.N)GO TO 220
210 CONTINUE
220 RETURN
    END
```

```
      SUBROUTINE IARNG(ITERM)
C*****************************************************************
C     THIS SUBROUTINE ARRANGES THE DEMAND POINTS IN EACH TERMINAL
C     AFTER BEING GROUPED BY SUBROUTINE IGRUP
C*****************************************************************
      COMMON IDIST(25,30),LDIST(25,30),KPR(25,30),IDEMND(25),
     *LDEMND(25),ICAP(9),NTRCK(9)
      COMMON ITER,ITYPE,N
      COMMON LINKS(50,26),LINKF(50,26),KD(50,25),KOK(50,25)
      COMMON JS(5,15),KOUNT(5),LD(25),LS(450),LF(450),B(25),
     *MF(200),MS(200),LEAD(25),IB(200),MR(25)
      COMMON IROUT(9,5),NCOMP(6),IIN(20,25),IRT(20,26),KL(26),
     *INCR(20,26),IRMIN(20,25),KLINK(26)
      COMMON JJGAP(50,25),LINKSN(26),LINKFN(26),L(26)
      K=KOUNT(ITERM)
      DO 530 I=1,K
      ITEMP=9999
      I1=I-1
      DO 520 IP=1,K
      IF(I.EQ.1)GO TO 505
      DO 500 II=1,I1
      IF(LD(II).EQ.JS(ITERM,IP))GO TO 520
  500 CONTINUE
  505 IF(JS(ITERM,IP).LT.ITEMP)GO TO 515
      GO TO 520
  515 ITEMP=JS(ITERM,IP)
  520 CONTINUE
      LD(I)=ITEMP
  530 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE ITREE(NTREE,NTK)
C*************************************************************
C     THIS SUBROUTINE FORMS MINIMUM 1-TREE(S).
C*************************************************************
      COMMON IDIST(25,30),LDIST(25,30),KPR(25,30),ICEMND(25),
     *LCEMND(25),ICAP(9),NTRCK(9)
      COMMON ITER,ITYPE,N
      COMMON LINKS(50,26),LINKF(50,26),KD(50,25),KOK(50,25)
      COMMON JS(5,15),KCUNT(5),LD(25),LS(450),LF(450),B(25),
     *MF(200),MS(200),LEAD(25),IB(200),MR(25)
      COMMON IRCUT(9,5),NCOMP(6),IIN(20,25),IRT(20,26),KL(26),
     *INCR(20,26),IRMIN(20,25),KLINK(26)
      COMMON JJGAP(50,25),LINKSN(26),LINKFN(26),L(26)
C
C     TO ARRANGE DISTANCES IN ASCENDING ORDER
C
      DO 1CCC NA=1,NTK
      ITEMP=9999
      DO 1C1C NB=2,N
      NC=NB-1
      DO 102C ND=1,NC
      IF(NA.EQ.1)GO TO 1035
      JPP=NA-1
      DO 1025 JP=1,JPP
      IF((ND.EQ.MF(JP)).AND.(NB.EQ.MS(JP)))GO TO 1020
 1025 CONTINUE
 1035 IF(IDIST(ND,NB).LE.ITEMP)GO TO 1030
      GO TO 1020
 1030 II=ND
      JJ=NB
      ITEMP=IDIST(ND,NB)
 1020 CONTINUE
 1010 CONTINUE
      MF(NA)=II
      MS(NA)=JJ
      IB(NA)=IDIST(II,JJ)
 1000 CONTINUE
      JC=1
      ICOUNT=1
      NTREE=1
      NN=1
      DO 104C K=1,NTK
      IF(JC.EQ.N)GO TO 1043
      ILC=JC-1
      GO TO 1042
 1043 IF(IDIST(MF(K),MS(K)).NE.IDIST(MF(K-1),MS(K-1)))GO TO 1220
      ILC=JC-2
 1042 IF(ICOUNT.GT.ILC)GO TO 1060
 1050 DO 107C M1=1,NTREE
      I=1
      KL(I)=0
      LEAD(I)=MS(K)
 1073 DO 1072 NP=1,ILC
      DO 1075 JI=1,I
      IF(KL(JI).EQ.NP)GO TO 1072
```

```
1075 CONTINUE
     DO 1080 JI=1,I
     IF(LEAD(JI).EQ.LINKS(M1,NP))GO TO 1090
1080 CONTINUE
     DO 1100 JJ=1,I
     IF(LEAD(JJ).EQ.LINKF(M1,NP))GO TO 1110
1100 CONTINUE
     GO TO 1072
1090 I=I+1
     KL(I)=NP
     LEAD(I)=LINKF(M1,NP)
     IF(LEAD(I).EQ.MF(K))GO TO 1150
     GO TO 1140
1110 I=I+1
     KL(I)=NP
     LEAD(I)=LINKS(M1,NP)
     IF(LEAD(I).EQ.MF(K))GO TO 1150
     GO TO 1140
1072 CONTINUE
     GO TO 1070
1140 GO TO 1073
1070 CONTINUE
1060 IF(ILC.NE.(JC-1))GO TO 1065
     DO 1180 NTNO=1,NTREE
     LINKS(NTNO,JC)=MF(K)
     LINKF(NTNO,JC)=MS(K)
1180 CONTINUE
     NCOUNT=1
     JC=JC+1
     GO TO 1040
1150 IF(IDIST(MF(K),MS(K)).EQ.IDIST(MF(K-1),MS(K-1)))GO TO1160
     GO TO 1040
1160 IF(ILC.NE.(JC-1))GO TO 1040
     ILC=ILC-1
     I=1
     LEAD(I)=MS(K)
     GO TO 1050
1065 GO TO (1066,1067,1079),NCOUNT
1079 NC=NTREE/3
     GO TO 1081
1067 NC=NTREE/2
1081 DO 1068 JB=1,NC
     LINKS((NTREE+JB),(JC-1))=MF(K)
     LINKF((NTREE+JB),(JC-1))=MS(K)
1068 CONTINUE
     DO 1069 JB=1,NC
     JCL=JC-2
     DO 1071 JP=1,JCL
     LINKS((NTREE+JB),JP)=LINKS(JB,JP)
     LINKF((NTREE+JB),JP)=LINKF(JB,JP)
1071 CONTINUE
1069 CONTINUE
     NCOUNT=NCOUNT+1
     NTREE=NTREE+NC
     NN=NTREE
```

```
      GO TO 1040
 1C66 NTREE=2*NTREE
      NCOUNT=NCOUNT+1
      N4=NTREE/2
      DO 119C NTT=1,N4
      LINKS((NTT+NN),(JC-1))=MF(K)
      LINKF((NTT+NN),(JC-1))=MS(K)
      JM=NTT+NN
      JCC=JC-1
 1190 CONTINUE
      NN=2*NN
      JC1=JC-2
      DO 123C NTT=1,N4
      DO 124C JKK=1,JC1
      LINKS((NTT+NN/2),JKK)=LINKS(NTT,JKK)
      LINKF((NTT+NN/2),JKK)=LINKF(NTT,JKK)
 1240 CONTINUE
 1230 CONTINUE
 1040 CONTINUE
C
C     TO ARRANGE TERMINAL DISTANCES IN ASCENDING ORDER
C
 1220 ICOUNT=0
      DO 125C KK=1,N
      ITEMP=9999
      DO 126C IK=1,N
      JK=KK-1
      IF(JK.EQ.ICOUNT)GO TO 1265
      DO 1225 K=1,JK
      IF(IK.EQ.MF(K))GO TO 1260
 1225 CONTINUE
 1265 IF(IDIST(IK,(N+1)).LE.ITEMP)GO TO 1270
      GO TO 1260
 1270 IKK=IK
      ITEMP=IDIST(IK,(N+1))
 1260 CONTINUE
      B(KK)=IDIST(IKK,(N+1))
      MF(KK)=IKK
      MS(KK)=N+1
 1250 CONTINUE
      DO 128C JT=1,NTREE
      LINKS(JT,N)=MF(1)
      LINKF(JT,N)=N+1
      LINKS(JT,(N+1))=MF(2)
      LINKF(JT,(N+1))=N+1
      N1=N+1
 1280 CONTINUE
      DO 129C KK=3,N
      IF(B(KK).EQ.B(KK-1))GO TO 1300
      GO TO 1310
 1300 DO 130C KP=3,KK
      IF(B(KP).NE.B(KP-1))GO TO 131C
 1305 CONTINUE
      NTREE=2*NTREE
      JTT=NTREE/2+1
```

```
      DO 1320 JT=JTT,NTREE
      LINKS(JT,N)=MF(1)
      LINKF(JT,N)=N+1
      LINKS(JT,(N+1))=MF(KK)
      LINKF(JT,(N+1))=N+1
 1320 CONTINUE
      NT=N-1
      DO 1330 NPP=1,NT
      DO 1340 JT=JTT,NTREE
      LINKS(JT,NPP)=LINKS((JT-NTREE/2),NPP)
      LINKF(JT,NPP)=LINKF((JT-NTREE/2),NPP)
 1340 CONTINUE
 1330 CONTINUE
 1290 CONTINUE
      NP=N+1
C
C     TO FIND OUT THE DEGREES OF NODE
C
      N1=N+1
 1310 DO 1350 NTR=1,NTREE
      DO 1360 JD=1,N
 1360 KOK(NTR,JD)=0
      DO 1370 JDD=1,N
      DO 1380 JL=1,N1
      IF(LINKS(NTR,JL).EQ.JDD)GO TO 1390
      IF(LINKF(NTR,JL).EQ.JDD)GO TO 1390
      GO TO 1380
 1390 KOK(NTR,JDD)=KOK(NTR,JDD)+1
 1380 CONTINUE
 1370 CONTINUE
 1350 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE KILTER(NTREE)
C*************************************************************
C     THIS SUBROUTINE FINDS THE NODES WHICH ARE OUT OF KILTER
C*************************************************************
      COMMON IDIST(25,30),LDIST(25,30),KPR(25,30),IDEMND(25),
     *LDEMND(25),ICAP(9),NTRCK(9)
      COMMON ITER,ITYPE,N
      COMMON LINKS(50,26),LINKF(50,26),KD(50,25),KOK(50,25)
      COMMON JS(5,15),KOUNT(5),LD(25),LS(450),LF(450),B(25),
     *MF(200),MS(200),LEAD(25),IB(200),MR(25)
      COMMON IROUT(9,5),NCOMP(6),IIN(20,25),IRT(20,26),KL(26),
     *INCR(20,26),IRMIN(20,25),KLINK(26)
      COMMON JJGAP(50,25),LINKSN(26),LINKFN(26),L(26)
      MM=0
      K=0
      IF(NTREE.EQ.1)GO TO 2700
      DO 2500 J=1,N
      DO 3000 I=2,NTREE
      IF(KOK(I,J).EQ.KOK((I-1),J))GO TO 3000
      GO TO 2500
 3000 CONTINUE
      DO 2510 II=1,NTREE
      KD(II,J)=KOK(II,J)-2
 2510 CONTINUE
      K=K+1
      MR(K)=J
      MM=MM+1
 2500 CONTINUE
      IF((MM-1))2700,2750,2750
 2700 DO 2000 II=1,NTREE
      DO 2010 JJ=1,N
 2010 KD(II,JJ)=KOK(II,JJ)-2
 2000 CONTINUE
      GO TO 3500
 2750 DO 2900 I=1,N
      DO 2910 J=1,NTREE
      DO 2920 KK=1,K
      IF(I.EQ.MR(KK))GO TO 2900
 2920 CONTINUE
      KD(J,I)=0
 2910 CONTINUE
 2900 CONTINUE
      DO 2911 I=1,NTREE
      DO 2912 J=1,N
      IF(KD(I,J).EQ.0)GO TO 2912
      GO TO 3500
 2912 CONTINUE
 2911 CONTINUE
      GO TO 2700
 3500 RETURN
      END
```

```
      SUBROUTINE IUPDAT(JNODE,JGAP)
C*************************************************************
C     THIS SUBROUTINE UPDATES THE DISTANCE MATRIX.
C*************************************************************
      COMMON IDIST(25,30),LDIST(25,30),KPR(25,30),IDEMND(25),
     *LDEMND(25),ICAP(9),NTRCK(9)
      COMMON ITER,ITYPE,N
      COMMON LINKS(50,26),LINKF(50,26),KD(50,25),KOK(50,25)
      COMMON JS(5,15),KOUNT(5),LD(25),LS(450),LF(450),B(25),
     *MF(200),MS(200),LEAD(25),IB(200),MR(25)
      COMMON IROUT(9,5),NCOMP(6),IIN(20,25),IRT(20,26),KL(26),
     *INCR(20,26),IRMIN(20,25),KLINK(26)
      COMMON JJGAP(50,25),LINKSN(26),LINKFN(26),L(26)
      IF(JNODE.EQ.1)GO TO 2000
      JNC=JNODE-1
      DO 2010 JJ=1,JNC
      IDIST(JJ,JNODE)=IDIST(JJ,JNODE)+JGAP
 2010 CONTINUE
 2000 JN=JNODE+1
      NT=N+1
      DO 2020 JJ=JN,NT
      IDIST(JNODE,JJ)=IDIST(JNODE,JJ)+JGAP
 2020 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE IGAPLO(NTREE,JGAPLO,JNODLO)
C*************************************************************
C     THIS SUBROUTINE FINDS THE MINIMUM GAP FOR UPDATING WHEN
C     NODE IS OUT-OF-KILTER LOW
C*************************************************************
      COMMON IDIST(25,30),LDIST(25,30),KPR(25,30),ICEMND(25),
     *LDEMND(25),ICAP(9),NTRCK(9)
      COMMON ITER,ITYPE,N
      COMMON LINKS(50,26),LINKF(50,26),KD(50,25),KOK(50,25)
      COMMON JS(5,15),KOUNT(5),LD(25),LS(450),LF(450),B(25),
     *MF(200),MS(200),LEAD(25),IB(200),MR(25)
      COMMON IRCLT(9,5),NCOMP(6),IIN(20,25),IRT(20,26),KL(26),
     *INCR(20,26),IRMIN(20,25),KLINK(26)
      COMMON JJGAP(50,25),LINKSN(26),LINKFN(26),L(26)
      ML=0
      JIGAP=9999
      NTCTL=NTREE*N
      N2=N-2
      N1=N-1
      DO 200 NTR=1,NTREE
      DO 250 I=1,N
      IF(KD(NTR,I).GE.0)GO TO 249
      IF(I.EQ.1)GO TO 3035
      I1=I-1
      DO 3030 II=1,I1
      DO 3040 JJ=I,N1
      IF(LINKS(NTR,JJ).EQ.I)GO TO 3040
      IF(LINKF(NTR,JJ).EQ.I)GO TO 3040
      JJGAP(NTR,I)=IDIST(II,I)-IDIST(LINKS(NTR,JJ),LINKF(NTR,JJ))
      IF(JJGAP(NTR,I).LE.0)GO TO 3040
C
C     CHECK IF THE NEW LINK FORMS A LOOP
C
      LINKSN(JJ)=II
      LINKFN(JJ)=I
      IF(JJ.EQ.1)GO TO 25
      JJ1=JJ-1
      DO 15 JD=1,JJ1
      LINKSN(JD)=LINKS(NTR,JD)
      LINKFN(JD)=LINKF(NTR,JD)
   15 CONTINUE
   25 JJ2=JJ+1
      DO 30 JD=JJ2,N1
      LINKSN(JD)=LINKS(NTR,JD)
      LINKFN(JD)=LINKF(NTR,JD)
   30 CONTINUE
      IJ=1
      KL(IJ)=JJ
      LEAD(IJ)=LINKFN(JJ)
   32 DO 35 NP=1,N1
      DO 38 JI=1,IJ
      IF(KL(JI).EQ.NP)GO TO 35
   38 CONTINUE
      DO 40 JI=1,IJ
      IF(LEAD(JI).EQ.LINKSN(NP))GO TO 45
```

```
   40 CONTINUE
      DO 50 JI=1,IJ
      IF(LEAD(JI).EQ.LINKFN(NP))GO TO 55
   50 CONTINUE
      GO TO 35
   45 IJ=IJ+1
      KL(IJ)=NP
      LEAD(IJ)=LINKFN(NP)
      DO 60 KK=1,IJ
      IF(LEAD(KK).EQ.LINKSN(JJ))GO TO 3040
   60 CONTINUE
      GO TO 39
   55 IJ=IJ+1
      KL(IJ)=NP
      LEAD(IJ)=LINKSN(NP)
      DO 70 KK=1,IJ
      IF(LEAD(KK).EQ.LINKSN(JJ))GO TO 3040
   70 CONTINUE
      GO TO 39
   35 CONTINUE
      GO TO 41
   39 GO TO 32
   41 IF(JJGAP(NTR,I).GT.JIGAP)GO TO 3040
      JIGAP=JJGAP(NTR,I)
      JNODLO=I
      JGAPLO=JIGAP
 3040 CONTINUE
 3030 CONTINUE
 3035 IF(I.EQ.N)GO TO 3000
      II=I+1
      DO 3020 II=I1,N
      DO 3060 JJ=1,N1
      IF(LINKS(NTR,JJ).EQ.I)GO TO 3060
      IF(LINKF(NTR,JJ).EQ.I)GO TO 3060
      JJGAP(NTR,I)=IDIST(I,II)-IDIST(LINKS(NTR,JJ),LINKF(NTR,JJ))
      IF(JJGAP(NTR,I).LE.0)GO TO 3060
C
C     CHECK IF LOOP IS FORMED
C
      LINKSN(JJ)=I
      LINKFN(JJ)=II
      IF(JJ.EQ.1)GO TO 3070
      JK1=JJ-1
      DO 3080 JK=1,JK1
      LINKSN(JK)=LINKS(NTR,JK)
      LINKFN(JK)=LINKF(NTR,JK)
 3080 CONTINUE
 3070 JK2=JJ+1
      DO 3090 JK=JK2,N
      LINKSN(JK)=LINKS(NTR,JK)
      LINKFN(JK)=LINKF(NTR,JK)
 3090 CONTINUE
      IJ=1
      KL(IJ)=JJ
      LEAD(IJ)=LINKFN(JJ)
```

```
3102 DO 3100 NP=1,N1
     DO 3105 JI=1,IJ
     IF(KL(JI).EQ.NP)GO TO 3100
3105 CONTINUE
     DO 3110 JI=1,IJ
     IF(LEAD(JI).EQ.LINKSN(NP))GO TO 3120
3110 CONTINUE
     DO 3130 JI=1,IJ
     IF(LEAD(JI).EQ.LINKFN(NP))GO TO 3140
3130 CONTINUE
     GO TO 3100
3120 IJ=IJ+1
     KL(IJ)=NP
     LEAD(IJ)=LINKFN(NP)
     DO 3150 KK=1,IJ
     IF(LEAD(KK).EQ.LINKSN(JJ))GO TO 3060
3150 CONTINUE
     GO TO 3107
3140 IJ=IJ+1
     KL(IJ)=NP
     LEAD(IJ)=LINKSN(NP)
     DO 3160 KK=1,IJ
     IF(LEAD(KK).EQ.LINKSN(JJ))GO TO 3060
3160 CONTINUE
     GO TO 3107
3100 CONTINUE
     GO TO 3111
3107 GO TO 3102
3111 IF(JJGAP(NTR,I).GT.JIGAP)GO TO 3060
     JIGAP=JJGAP(NTR,I)
     JNCDLC=I
     JGAPLC=JIGAP
3060 CONTINUE
3020 CONTINUE
     JJGAP(NTR,I)=IDIST(I,(N+1))-IDIST(LINKS(NTR,(N+1)),LINKF(NTR,(N+1)
    1))
     IF(JJGAP(NTR,I).LE.0.OR.JJGAP(NTR,I).GE.JIGAP)GO TO 250
     JIGAP=JJGAP(NTR,I)
     JNCDLC=I
     JGAPLO=JIGAP
     GO TO 250
 249 ML=ML+1
     IF(ML.LT.NTOTL)GO TO 250
     JNCDLO=N
     JGAPLO=9999
 250 CONTINUE
 200 CONTINUE
3000 RETURN
     END
```

```
      SUBROUTINE IGAPHI(NTREE,JGAPHI,JNODHI)
C*************************************************************
C     THIS SUBROUTINE FINDS THE MINIMUM GAP FOR UPDATING WHEN
C     NODE IS OUT-OF-KILTER HIGH
C*************************************************************
      COMMON IDIST(25,30),LDIST(25,30),KPR(25,30),IDEMND(25),
     *LDEMND(25),ICAP(9),NTRCK(9)
      COMMON ITER,ITYPE,N
      COMMON LINKS(50,26),LINKF(50,26),KD(50,25),KOK(50,25)
      COMMON JS(5,15),KOUNT(5),LD(25),LS(450),LF(450),B(25),
     *MF(200),MS(200),LEAD(25),IB(200),MR(25)
      COMMON IRCUT(9,5),NCOMP(6),IIN(20,25),IRT(20,26),KL(26),
     *INCR(20,26),IRMIN(20,25),KLINK(26)
      COMMON JJGAP(50,25),LINKSN(26),LINKFN(26),L(26)
      JIGAP=9999
      N1=N-1
      ML=0
      NTOTL=NTREE*N
      DO 2900 NTR=1,NTREE
      DO 3000 I=1,N
      IF(KD(NTR,I).LE.0)GO TO 2999
      K=0
      DO 3155 JP=1,N1
      IF(LINKF(NTR,JP).EQ.I)GO TO 3170
      IF(LINKS(NTR,JP).EQ.I)GO TO 3165
      GO TO 3155
 3170 K=K+1
      L(K)=LINKS(NTR,JP)
      GO TO 3155
 3165 K=K+1
      L(K)=LINKF(NTR,JP)
 3155 CONTINUE
      DO 3200 JK=1,K
      DO 3210 JJ=2,N
      I1=JJ-1
      DO 3220 II=1,I1
      IF(L(JK).EQ.(N+1))GO TO 3200
      IF(II.EQ.I.OR.JJ.EQ.I)GO TO 3220
C     CHECK WHETHER THE LINK CONSIDERED IS FROM TREE OR NOT
      DO 3240 M=1,N1
      IF(LINKS(NTR,M).EQ.II.AND.LINKF(NTR,M).EQ.JJ)GO TO 3220
 3240 CONTINUE
      IF(L(JK).GT.I)GO TO 3250
      JJGAP(NTR,I)=IDIST(II,JJ)-IDIST(L(JK),I)
      GO TO 3260
 3250 JJGAP(NTR,I)=IDIST(II,JJ)-IDIST(I,L(JK))
 3260 IF(JJGAP(NTR,I).LE.0)GO TO 3220
C     TO CHECK IF THE NEW LINK FORMS A LOOP
      DO 3300 JM=1,N1
      IF(LINKS(NTR,JM).EQ.L(JK))GO TO 3310
      IF(LINKF(NTR,JM).EQ.L(JK))GO TO 3320
      GO TO 3330
 3310 IF(LINKF(NTR,JM).NE.I)GO TO 3330
      LINKSN(JM)=II
      LINKFN(JM)=JJ
```

```
      KR=JM
      GO TO 3300
3320  IF(LINKS(NTR,JM).NE.I)GO TO 3330
      LINKSN(JM)=II
      LINKFN(JM)=JJ
      KR=JM
      GO TO 3300
3330  LINKSN(JM)=LINKS(NTR,JM)
      LINKFN(JM)=LINKF(NTR,JM)
3300  CONTINUE
      IJ=1
      KL(IJ)=KR
      LEAD(IJ)=LINKFN(KR)
3333  DO 3350 NP=1,N1
      DO 3355 JI=1,IJ
      IF(KL(JI).EQ.NP)GO TO 3350
3355  CONTINUE
      DO 3360 JI=1,IJ
      IF(LEAD(JI).EQ.LINKSN(NP))GO TO 3380
3360  CONTINUE
      DO 3370 JI=1,IJ
      IF(LEAD(JI).EQ.LINKFN(NP))GO TO 3390
3370  CONTINUE
      GO TO 3350
3380  IJ=IJ+1
      KL(IJ)=NP
      LEAD(IJ)=LINKFN(NP)
      DO 3400 KK=1,IJ
      IF(LEAD(KK).EQ.LINKSN(KR))GO TO 3220
3400  CONTINUE
      GO TO 3331
3390  IJ=IJ+1
      KL(IJ)=NP
      LEAD(IJ)=LINKSN(NP)
      DO 3410 KK=1,IJ
      IF(LEAD(KK).EQ.LINKSN(KR))GO TO 3220
3410  CONTINUE
      GO TO 3331
3350  CONTINUE
      GO TO 3332
3331  GO TO 3333
3332  IF(JJGAP(NTR,I).GT.JIGAP)GO TO 3220
      JIGAP=JJGAP(NTR,I)
      JNODHI=I
      JGAPHI=JIGAP
3220  CONTINUE
3210  CONTINUE
3200  CONTINUE
      IF(LINKS(NTR,N).NE.I.OR.LINKS(NTR,(N+1)).NE.I)GO TO 3000
      DO 2998 JP=1,N
      IF(LINKS(NTR,N).EQ.JP.OR.LINKS(NTR,(N+1)).EQ.JP)GO TO 2998
      JJGAP(NTR,I)=IDIST(JP,(N+1))-IDIST(I,(N+1))
      IF(JJGAP(NTR,I).LE.0.OR.JJGAP(NTR,I).GT.JIGAP)GO TO 2998
      JIGAP=JJGAP(NTR,I)
      JGAPHI=JIGAP
```

```
      JNCDHI=I
2998  CONTINUE
      GO TO 3000
2999  ML=ML+1
      IF(ML.LT.NTOTL)GO TO 3000
      JNCDHI=N
      JGAPHI=9999
3000  CONTINUE
2900  CONTINUE
      RETURN
      END
```

```
      SUBROUTINE IROUTS(JE,NM)
C*****************************************************************
C     THIS SUBROUTINE FORMS ROUTES BY APPLYING CUTS AT THE
C     ARC WITH MINIMUM INCREMENT
C*****************************************************************
      COMMON IDIST(25,30),LDIST(25,30),KPR(25,30),IDEMND(25),
     *LDEMND(25),ICAP(9),NTRCK(9)
      COMMON ITER,ITYPE,N
      COMMON LINKS(50,26),LINKF(50,26),KD(50,25),KOK(50,25)
      COMMON JS(5,15),KOUNT(5),LD(25),LS(450),LF(450),B(25),
     *MF(200),MS(200),LEAD(25),IB(200),MR(25)
      COMMON IROUT(9,5),NCOMP(6),IIN(20,25),IRT(20,26),KL(26),
     *INCR(20,26),IRMIN(20,25),KLINK(26)
      COMMON JJGAP(50,25),LINKSN(26),LINKFN(26),L(26)
      NM=1
      K=1
      KL(K)=0
      N1=N-1
      IRT(1,1)=LINKS(JE,N)
      IRT(1,N)=LINKS(JE,(N+1))
 6060 DO 6000 I=1,N1
      DO 6005 J=1,K
      IF(KL(J).EQ.I)GO TO 6000
 6005 CONTINUE
      IF(LINKS(JE,I).EQ.IRT(1,K))GO TO 6010
      IF(LINKF(JE,I).EQ.IRT(1,K))GO TO 6020
      GO TO 6000
 6010 K=K+1
      IRT(1,K)=LINKF(JE,I)
      KL(K)=I
      GO TO 6050
 6020 K=K+1
      IRT(1,K)=LINKS(JE,I)
      KL(K)=I
      GO TO 6050
 6000 CONTINUE
      GO TO 6090
 6050 GO TO 6060
 6090 DO 6080 I=1,N1
      IF(IRT(1,I).GT.IRT(1,(I+1)))GO TO 6091
      INCR(1,I)=IDIST(IRT(1,I),(N+1))+IDIST(IRT(1,(I+1)),(N+1))-IDIST(I
     1RT(1,I),IRT(1,(I+1)))
      GO TO 6080
 6091 INCR(1,I)=IDIST(IRT(1,I),(N+1))+IDIST(IRT(1,(I+1)),(N+1))-IDIST(I
     1RT(1,(I+1)),IRT(1,I))
 6080 CONTINUE
      ITRCK=0
      DO 6081 I=1,N
 6081 ITRCK=ITRCK+IDEMND(I)
      IF(ITRCK.LE.ICAP(1))GO TO 6082
      GO TO 6095
 6082 IF(NTRCK(1).EQ.0.AND.ITRCK.GT.ICAP(2))GO TO 6095
      DO 6096 JB=1,ITYPE
      IF(ITRCK.LE.ICAP(JB))GO TO 6096
      NTRCK(JB-1)=NTRCK(JB-1)-1
```

```
      GO TO 6C98
 6096 CONTINUE
      NTRCK(ITYPE)=NTRCK(ITYPE)-1
 6C98 NM=1
      DO 6C99 I=1,N
 6099 IRCUT(NM,I)=IRT(NM,I)
      NCOMP(NM)=N
      GO TO 6900
C
C     ON TWO CONDITIONS TOUR WILL BE BROKEN INTO ROUTES
C   1 BREAK AT THE MINIMUM INCREMENT IF TRUCK CAN BE ASSIGNED
C     TO EACH SUBLINK
C   2 MINIMUM INCREASE SHOULD BREAK UP THE TOTAL TOUR IN
C     ALMOST EQUAL HALVES
C
 6C95 LM=2
      JCOMP=C
      I=1
      KLINK(1)=N1
 7C00 DO 16C1 K=1,I
      IF(KLINK(K).EQ.0)GO TO 1601
      K1=KLINK(K)
      K2=K1+1
      CALL IASCND(K1,K)
      N1=NM
      M=1
      IF(IRMIN(K,1).EQ.1)GO TO 6150
      IF(IRMIN(K,1).EQ.K1)GO TO 6160
      GO TO 6170
 6150 CALL IFIRST(K,K1,JCOMP,NM,M)
      IF(NM.NE.(N1+2))GO TO 6158
      LTEST=NCOMP(NM-1)+NCOMP(NM-2)
      IF(LTEST.EQ.K2)GO TO 1601
 6158 IF(IRMIN(K,2).EQ.K1)GO TO 6160
      GO TO 6170
 6160 CALL ISECND(K,K1,JCOMP,NM,M)
      IF(NM.NE.(N1+2))GO TO 6159
      LTEST=NCOMP(NM-1)+NCOMP(NM-2)
      IF(LTEST.EQ.K2)GO TO 1601
 6159 IF(IRMIN(K,2).EQ.1)GO TO 6150
      GO TO 617C
 6170 CALL IHALF(K,K1,M,JCOMP,NM,LM)
      IF(JCOMP.LT.N)GO TO 6520
      GO TO 69CC
 6520 IF(K.EQ.I)GO TO 6530
 1601 CONTINUE
      IF(JCOMP.EQ.N)GO TO 6900
 6530 DO 654C LL=1,I
 6540 KLINK(LL)=0
      I=LM-1
      GO TO 7C0C
 6900 RETURN
      END
```

```
      SUBROUTINE IFIRST(K,K1,JCOMP,NM,M)
C*****************************************************************
C     THIS SUBROUTINE FORMS FEASIBLE SUBTOURS FOR THE ARC
C     WITH MINIMUM INCREMENT
C*****************************************************************
      COMMON IDIST(25,30),LDIST(25,30),KPR(25,30),IDEMND(25),
     *LDEMND(25),ICAP(9),NTRCK(9)
      COMMON ITER,ITYPE,N
      COMMON LINKS(50,26),LINKF(50,26),KD(50,25),KOK(50,25)
      COMMON JS(5,15),KOUNT(5),LD(25),LS(450),LF(450),B(25),
     *MF(200),MS(200),LEAD(25),IB(200),MR(25)
      COMMON IROUT(9,5),NCOMP(6),IIN(20,25),IRT(20,26),KL(26),
     *INCR(20,26),IRMIN(20,25),KLINK(26)
      COMMON JJGAP(50,25),LINKSN(26),LINKFN(26),L(26)
      NCOUNT=0
      ITRCK=0
      K2=K1+1
      M=M+1
      DO 6165 JP=2,K2
      ITRCK=ITRCK+IDEMND(IRT(K,JP))
 6165 CONTINUE
      IF(ITRCK.LE.ICAP(1))GO TO 6190
      GO TO 6251
 6190 IF(NTRCK(1).EQ.0.AND.ITRCK.LE.ICAP(2))GO TO 6251
      IROUT(NM,1)=IRT(K,1)
      NCOMP(NM)=1
      NM=NM+1
      JCOMP=JCOMP+1
      DO 6201 JB=1,ITYPE
      IF(IDEMND(IRT(K,1)).LE.ICAP(JB))GO TO 6201
      NTRCK(JB-1)=NTRCK(JB-1)-1
      GO TO 6202
 6201 CONTINUE
      NTRCK(ITYPE)=NTRCK(ITYPE)-1
 6202 DO 6200 J=1,K1
      IROUT(NM,J)=IRT(K,(J+1))
      JCOMP=JCOMP+1
      NCOUNT=NCOUNT+1
 6200 CONTINUE
      NCOMP(NM)=NCOUNT
      DO 6204 JB=1,ITYPE
      IF(ITRCK.LE.ICAP(JB))GO TO 6204
      NTRCK(JB-1)=NTRCK(JB-1)-1
      GO TO 6206
 6204 CONTINUE
      NTRCK(ITYPE)=NTRCK(ITYPE)-1
 6206 NM=NM+1
 6251 RETURN
      END
```

```
      SUBROUTINE ISECND(K,K1,JCOMP,NM,M)
C*****************************************************************
C     THIS SUBROUTINE FORMS FEASIBLE SUBTOURS FOR THE ARC
C     WITH NEXT MINIMUM INCREMENT
C*****************************************************************
      COMMON IDIST(25,30),LDIST(25,30),KPR(25,30),IDEMND(25),
     *LDEMND(25),ICAP(9),NTRCK(9)
      COMMON ITER,ITYPE,N
      COMMON LINKS(50,26),LINKF(50,26),KD(50,25),KOK(50,25)
      COMMON JS(5,15),KOUNT(5),LD(25),LS(450),LF(450),B(25),
     *MF(200),MS(200),LEAD(25),IB(200),MR(25)
      COMMON IROUT(9,5),NCOMP(6),IIN(20,25),IRT(20,26),KL(26),
     *INCR(20,26),IRMIN(20,25),KLINK(26)
      COMMON JJGAP(50,25),LINKSN(26),LINKFN(26),L(26)
      NCOUNT=0
      M=M+1
      ITRCK=0
      DO 6195 JP=1,K1
      ITRCK=ITRCK+IDEMND(IRT(K,JP))
 6195 CONTINUE
      IF(ITRCK.LE.ICAP(1))GO TO 6210
      GO TO 6225
 6210 IF(NTRCK(1).EQ.0.AND.ITRCK.GT.ICAP(2))GO TO 6225
      IROUT(NM,1)=IRT(K,(K1+1))
      NCOMP(NM)=1
      NM=NM+1
      JCOMP=JCOMP+1
      DO 6211 JB=1,ITYPE
      IF(IDEMND(IRT(K,(K1+1))).LE.ICAP(JB))GO TO 6211
      NTRCK(JB-1)=NTRCK(JB-1)-1
      GO TO 6212
 6211 CONTINUE
      NTRCK(ITYPE)=NTRCK(ITYPE)-1
 6212 DO 6220 J=1,K1
      IROUT(NM,J)=IRT(K,J)
      JCOMP=JCOMP+1
      NCOUNT=NCOUNT+1
 6220 CONTINUE
      NCOMP(NM)=NCOUNT
      NM=NM+1
      DO 6221 JB=1,ITYPE
      IF(ITRCK.LE.ICAP(JB))GO TO 6221
      NTRCK(JB-1)=NTRCK(JB-1)-1
      GO TO 6225
 6221 CONTINUE
      NTRCK(ITYPE)=NTRCK(ITYPE)-1
 6225 RETURN
      END
```

```
      SUBROUTINE IHALF(K,K1,M,JCOMP,NM,LM)
C*************************************************************
C     THIS SUBROUTINE FORMS NON-FEASIBLE SUBTOURS FOR
C     CONSIDERATION FOR FURTHER CUTS WHEN FEASIBLE SUBTOURS
C     ARE NOT POSSIBLE
C*************************************************************
      COMMON IDIST(25,30),LDIST(25,30),KPR(25,30),IDEMND(25),
     *LDEMND(25),ICAP(9),NTRCK(9)
      COMMON ITER,ITYPE,N
      COMMON LINKS(50,26),LINKF(50,26),KD(50,25),KOK(50,25)
      COMMON JS(5,15),KOUNT(5),LC(25),LS(450),LF(450),B(25),
     *MF(200),MS(200),LEAD(25),IB(200),MR(25)
      COMMON IROUT(9,5),NCOMP(6),IIN(20,25),IRT(20,26),KL(26),
     *INCR(20,26),IRMIN(20,25),KLINK(26)
      COMMON JJGAP(50,25),LINKSN(26),LINKFN(26),L(26)
      LP=0
      ML=0
      A1=IRMIN(K,M)
      A2=IRMIN(K,(M+1))
      K2=K1+1
      P=K2
      AB1=ABS(A1-P/2)
      AB2=ABS(A2-P/2)
      GO TO 6280
 6289 NM=NM-NCHECK
      JCOMP=JCOMP-JCHECK
      IF(JCHECK.EQ.0)GO TO 6302
      DO 6301 IP=1,NCHECK
      DO 6301 IC=1,JCHECK
      IROUT((NM+IP),IC)=0
 6301 CONTINUE
 6302 ITRCK1=0
      IRP=IRMIN(K,(M+1))
      DO 6290 J=1,IRP
      ITRCK1=ITRCK1+IDEMND(IRT(K,J))
 6290 CONTINUE
      ITRCK2=0
      MK=IRMIN(K,(M+1))+1
      DO 6300 J=MK,K2
      ITRCK2=ITRCK2+IDEMND(IRT(K,J))
 6300 CONTINUE
      IF(ITRCK1.LE.ICAP(1))GO TO 6310
      ML=1
      IF(ITRCK2.LE.ICAP(1))GO TO 6325
      IF(AB2-AB1)6325,6421,6421
 6310 IF(NTRCK(1).EQ.0.AND.ITRCK1.GT.ICAP(2))GO TO 6325
      NCOUNT=0
      DO 6340 J=1,IRP
      IROUT(NM,J)=IRT(K,J)
      JCOMP=JCOMP+1
      NCOUNT=NCOUNT+1
 6340 CONTINUE
      NCOMP(NM)=NCOUNT
      NM=NM+1
      DO 6341 JB=1,ITYPE
```

```
      IF(ITRCK1.LE.ICAP(JB))GO TO 6341
      NTRCK(JB-1)=NTRCK(JB-1)-1
      GO TO 6342
 6341 CONTINUE
      NTRCK(ITYPE)=NTRCK(ITYPE)-1
 6342 IF(ITRCK2.LE.ICAP(1))GO TO 6320
      LP=1
      GO TO 6315
 6320 IF(NTRCK(1).EQ.0.AND.ITRCK2.GT.ICAP(2))GO TO 6315
      NCOUNT=0
      DO 6345 J=MK,K2
      IRCUT(NM,J)=IRT(K,J)
      JCOMP=JCOMP+1
      NCOUNT=NCOUNT+1
 6345 CONTINUE
      NCOMP(NM)=NCOUNT
      DO 6346 JB=1,ITYPE
      IF(ITRCK2.LE.ICAP(JB))GO TO 6346
      NTRCK(JB-1)=NTRCK(JB-1)-1
      GO TO 6347
 6346 CONTINUE
      NTRCK(ITYPE)=NTRCK(ITYPE)-1
 6347 GO TO 1600
 6315 IF(LP.EQ.1)GO TO 6316
      LM=LM-1
      IF(AB1.LE.AB2)GO TO 6421
      LM=LM+1
 6316 DO 6349 J=MK,K1
      J1=J-MK+1
      IRT(LM,J1)=IRT(K,J)
      INCR(LM,J1)=INCR(K,J)
 6349 CONTINUE
      IRT(LM,(K2-MK+1))=IRT(K,K2)
      KLINK(LM)=K1-MK+1
      LM=LM+1
      GO TO 1600
 6325 IF(AB1.LE.AB2.AND.ITRCK2.GT.ICAP(1))GO TO 6421
      IR2=IRMIN(K,(M+1))-1
      DO 6350 J=1,IR2
      IRT(LM,J)=IRT(K,J)
      INCR(LM,J)=INCR(K,J)
 6350 CONTINUE
      IRT(LM,IRMIN(K,(M+1)))=IRT(K,IRMIN(K,(M+1)))
      KLINK(LM)=IR2
      LM=LM+1
      IF(ITRCK2.LE.ICAP(1))GO TO 6320
      GO TO 6315
 6280 ITRCK1=0
      NCHECK=0
      JCHECK=0
      IR1=IRMIN(K,M)
      DO 6380 J=1,IR1
      ITRCK1=ITRCK1+ICEMND(IRT(K,J))
 6380 CONTINUE
      IR2=IRMIN(K,M)+1
```

```
      ITRCK2=0
      DO 645C J=IR2,K2
      ITRCK2=ITRCK2+IDEMND(IRT(K,J))
 6450 CONTINUE
      IF(ITRCK1.LE.ICAP(1))GO TO 6390
      IF(M.EQ.1)GO TO 6421
 6420 IF(ITRCK2.LE.ICAP(1))GO TO 6410
      GO TO 6289
 6421 IR=IRMIN(K,M)-1
      DO 64CC J=1,IR
      IRT(LM,J)=IRT(K,J)
      INCR(LM,J)=INCR(K,J)
 6400 CONTINUE
      IRT(LM,IRMIN(K,M))=IRT(K,IRMIN(K,M))
      KLINK(LM)=IR
      LM=LM+1
      IF(ML.GT.1)GO TO 1600
      ML=1
      GO TO 6410
 6390 IF(NTRCK(1).EQ.C.AND.ITRCK1.GT.ICAP(2))GO TO 6420
      MJ=IRMIN(K,M)
      DO 643C J=1,MJ
      IRCUT(NM,J)=IRT(K,J)
      JCOMP=JCOMP+1
      JCHECK=JCHECK+1
 6430 CONTINUE
      NCOMP(NM)=MJ
      NCHECK=NCHECK+1
      ML=1
      NM=NM+1
      DO 6431 JB=1,ITYPE
      IF(ITRCK1.LE.ICAP(JB))GO TO 6431
      NTRCK(JB-1)=NTRCK(JB-1)-1
      GO TO 641C
 6431 CONTINUE
      NTRCK(ITYPE)=NTRCK(ITYPE)-1
 6410 IF(ITRCK2.LE.ICAP(1))GO TO 6460
      IF(M.EQ.1)GO TO 6456
 6455 IF(ML.EQ.1)GO TO 6456
      GO TO 6289
 6456 DO 647C J=IR2,K1
      IRT(LM,(J-IRMIN(K,M)))=IRT(K,J)
      INCR(LM,(J-IRMIN(K,M)))=INCR(K,J)
 6470 CONTINUE
      IRT(LM,(K2-IRMIN(K,M)))=IRT(K,K2)
      KLINK(LM)=K1-IR2+1
      LM=LM+1
      GO TO 16CC
 6460 IF(NTRCK(1).EQ.0.AND.ITRCK2.GT.ICAP(2))GO TO 6455
      DO 65CC J=IR2,K2
      IRCUT(NM,(J-IRMIN(K,M)))=IRT(K,J)
      JCOMP=JCOMP+1
      JCHECK=JCHECK+1
 6500 CONTINUE
      NCOMP(NM)=K2-IR2+1
```

```
      NCHECK=NCHECK+1
      NM=NM+1
      DO 6501 JB=1,ITYPE
      IF(ITRCK2.LE.ICAP(JB))GO TO 6501
      NTRCK(JB-1)=NTRCK(JB-1)-1
      IF(ML.EQ.1)GO TO 1600
      ML=ML+2
      GO TO 6421
 6501 CONTINUE
      NTRCK(ITYPE)=NTRCK(ITYPE)-1
      IF(ML.EQ.1)GO TO 1600
      ML=ML+2
      GO TO 6421
 1600 RETURN
      END
```

```
      SUBROUTINE IASCND(K1,K)
C*****************************************************************
C     THIS SUBROUTINE ARRANGES THE INCREMENTS DUE TO CUT IN
C     ASCENDING ORDER TO FACILITATE THE CUT
C*****************************************************************
      COMMON IDIST(25,30),LDIST(25,30),KPR(25,30),IDEMND(25),
     *LDEMND(25),ICAP(9),NTRCK(9)
      COMMON ITER,ITYPE,N
      COMMON LINKS(50,26),LINKF(50,26),KD(50,25),KOK(50,25)
      COMMON JS(5,15),KOUNT(5),LC(25),LS(450),LF(450),B(25),
     *MF(200),MS(200),LEAD(25),IB(200),MR(25)
      COMMON IROUT(9,5),NCOMP(6),IIN(20,25),IRT(20,26),KL(26),
     *INCR(20,26),IRMIN(20,25),KLINK(26)
      COMMON JJGAP(50,25),LINKSN(26),LINKFN(26),L(26)
      DO 6100 IP=1,K1
      ITEMP=9999
      DO 6110 J=1,K1
      IF(IP.EQ.1)GO TO 6130
      I1=IP-1
      DO 6120 JK=1,I1
      IF(IRMIN(K,JK).EQ.J)GO TO 6110
 6120 CONTINUE
 6130 IF(INCR(K,J)-ITEMP)6140,6110,6110
 6140 ITEMP=INCR(K,J)
      II=J
 6110 CONTINUE
      IRMIN(K,IP)=II
      IIN(K,IP)=INCR(K,II)
 6100 CONTINUE
      RETURN
      END
```

APPENDIX B

SINGLE-TERMINAL PROBLEMS

PROBLEM 1

NUMBER OF DEMAND POINTS = 6 NUMBER OF TERMINALS = 1

DEMAND POINT	DEMAND AT THE POINT	DISTANCE TO THE TERMINAL
1	12	107
2	8	224
3	15	154
4	6	108
5	14	113
6	8	207

TRUCKS

CAPACITY	NUMBER
4C	1
3C	2
2C	2

DISTANCE MATRIX

(2)	331				
(3)	152	284			
(4)	172	167	121		
(5)	217	125	216	114	
(6)	132	388	110	221	317

FINAL SCLCTICN

NC.	TERMINAL	ROUTES	LCAD	DISTANCE
1	1	T1 5 2 4 T1	28	513
2	1	T1 3 6 1 T1	35	503

TCTAL CISTANCE = 1016

TRUCK AVAILABILITY AND ALLCCATICN TABLE

TRUCK	40	30	20
AVAILABLE	1	2	2
ALLOCATED	1	1	C

PROBLEM 2

NUMBER OF DEMAND POINTS = 6 NUMBER OF TERMINALS = 1

DEMAND POINT	DEMAND AT THE POINT	DISTANCE TO THE TERMINAL
1	7	38
2	9	42
3	8	56
4	7	63
5	6	64
6	5	80

TRUCKS

CAPACITY	NUMBER
18	1
15	1
12	2

DISTANCE MATRIX

(2)	35				
(3)	22	33			
(4)	54	22	45		
(5)	38	27	20	29	
(6)	62	38	65	22	45

FINAL SOLUTION

NO.	TERMINAL	ROUTES	LOAD	DISTANCE
1	1	T1 1 3 T1	15	116
2	1	T1 5 6 4 T1	18	194
3	1	T1 2 T1	9	84

TOTAL DISTANCE = 394

TRUCK AVAILABILITY AND ALLOCATION TABLE

TRUCK	18	15	12
AVAILABLE	1	1	2
ALLOCATED	1	1	1

PROBLEM 3

NUMBER OF DEMAND POINTS = 6 NUMBER OF TERMINALS = 1

DEMAND POINT	DEMAND AT THE POINT	DISTANCE TO THE TERMINAL
1	35	51
2	34	112
3	17	40
4	23	25
5	16	22
6	28	55

TRUCKS

CAPACITY	NUMBER
100	1
80	1
60	2

DISTANCE MATRIX

(2)	124				
(3)	53	83			
(4)	26	122	40		
(5)	58	90	31	32	
(6)	78	58	25	65	33

FINAL SCLLTICN

NC.	TERMINAL	ROUTES	LCAC	DISTANCE
1	1	T1 4 1 T1	58	1C2
2	1	T1 3 6 2 5 11	95	235

TCTAL DISTANCE = 337

TRUCK AVAILABILITY AND ALLCCATICN TABLE

TRLCK	10C	PC	6C
AVAILABLE	1	1	2
ALLCCATEC	1	C	1

PROBLEM 4

NUMBER OF DEMAND POINTS = 6 NUMBER OF TERMINALS = 1

DEMAND POINT	DEMAND AT THE POINT	DISTANCE TO THE TERMINAL
1	20	18
2	16	67
3	24	41
4	15	58
5	32	54
6	22	71

TRUCKS

CAPACITY	NUMBER
80	1
70	1
60	2

DISTANCE MATRIX

(2)	49				
(3)	50	98			
(4)	40	21	89		
(5)	36	39	64	30	
(6)	60	74	30	65	35

FINAL SOLUTION

NO.	TERMINAL	ROUTES					LOAD	DISTANCE
1	1	T1	1	4	2	T1	51	146
2	1	T1	5	6	3	T1	78	160

TOTAL DISTANCE = 306

TRUCK AVAILABILITY AND ALLOCATION TABLE

TRUCK	80	70	60
AVAILABLE	1	1	2
ALLOCATED	1	0	1

PROBLEM 5

NUMBER OF DEMAND POINTS = 6 NUMBER OF TERMINALS = 1

DEMAND POINT	DEMAND AT THE POINT	DISTANCE TO THE TERMINAL
1	25	252
2	18	187
3	17	497
4	16	346
5	17	209
6	28	297

TRUCKS

CAPACITY	NUMBER
90	1
60	2
50	1

DISTANCE MATRIX

(2)	154				
(3)	215	367			
(4)	505	386	720		
(5)	366	247	581	137	
(6)	173	94	378	478	339

FINAL SCLLTICN

NC.	TERMINAL	ROUTES	LCAD	DISTANCE
1	1	T1 2 6 3 1 T1	88	1126
2	1	T1 4 5 T1	33	692

TCTAL DISTANCE = 1818

TRUCK AVAILABILITY AND ALLCCATION TABLE

TRLCK	90	60	5C
AVAILABLE	1	2	1
ALLOCATED	1	C	1

PROBLEM 6

NUMBER OF DEMAND POINTS = 6 NUMBER OF TERMINALS = 1

DEMAND POINT	DEMAND AT THE POINT	DISTANCE TO THE TERMINAL
1	15	265
2	9	154
3	16	108
4	10	326
5	7	113
6	11	207

TRUCKS

CAPACITY	NUMBER
50	1
40	1
30	2

DISTANCE MATRIX

(2)	160				
(3)	277	121			
(4)	86	131	248		
(5)	375	216	114	347	
(6)	61	110	221	119	317

FINAL SCLLTICN

NC.	TERMINAL	ROUTES	LCAC	DISTANCE
1	1	T1 2 4 1 6 T1	45	639
2	1	T1 3 5 T1	23	335

TOTAL DISTANCE = 974

TRLCK AVAILABILITY ANC ALLCCATION TABLE

TRLCK	5C	4C	3C
AVAILABLE	1	1	2
ALLCCATEC	1	C	1

PROBLEM 7

NUMBER OF DEMAND POINTS = 6 NUMBER OF TERMINALS = 1

DEMAND POINT	DEMAND AT THE POINT	DISTANCE TO THE TERMINAL
1	20	432
2	15	366
3	16	552
4	7	508
5	9	389
6	17	180

TRUCKS

CAPACITY	NUMBER
50	1
40	1
30	2

DISTANCE MATRIX

(2)	154				
(3)	400	248			
(4)	505	386	256		
(5)	366	247	210	137	
(6)	252	187	375	346	209

FINAL SOLUTION

NO.	TERMINAL	ROUTES	LOAD	DISTANCE
1	1	T1 6 1 T1	37	864
2	1	T1 5 4 3 2 T1	47	1396

TOTAL DISTANCE = 2260

TRUCK AVAILABILITY AND ALLOCATION TABLE

TRUCK	5C	4C	3C
AVAILABLE	1	1	2
ALLOCATED	1	1	0

PROBLEM 8

NUMBER OF DEMAND POINTS = 7 NUMBER OF TERMINALS = 1

DEMAND POINT	DEMAND AT THE POINT	DISTANCE TO THE TERMINAL
1	24	107
2	15	137
3	16	331
4	34	152
5	18	172
6	24	217
7	25	132

TRUCKS

CAPACITY	NUMBER
90	1
80	1
70	2

DISTANCE MATRIX

(2)	244					
(3)	224	468				
(4)	154	212	284			
(5)	108	309	167	121		
(6)	113	354	125	216	114	
(7)	207	110	388	110	221	317

FINAL SOLUTION

NO.	TERMINAL	ROUTES	LOAD	DISTANCE
1	1	T1 2 7 4 T1	74	509
2	1	T1 5 3 6 1 T1	82	684

TOTAL DISTANCE = 1193

TRUCK AVAILABILITY AND ALLOCATION TABLE

TRUCK	90	80	70
AVAILABLE	1	1	2
ALLOCATED	1	1	0

PROBLEM 9

NUMBER OF DEMAND POINTS = 8 NUMBER OF TERMINALS = 1

DEMAND POINT	DEMAND AT THE POINT	DISTANCE TO THE TERMINAL
1	36	151
2	23	252
3	30	250
4	27	350
5	23	372
6	22	170
7	18	70
8	40	390

TRUCKS

CAPACITY	NUMBER
100	1
90	1
80	2
50	1
30	1

DISTANCE MATRIX

(2)	338						
(3)	391	272					
(4)	499	201	146				
(5)	476	145	349	205			
(6)	288	336	130	253	454		
(7)	193	290	200	309	410	95	
(8)	511	344	144	176	381	220	321

FINAL SCLLTICN

NC.	TERMINAL	ROUTES	LCAD	DISTANCE
1	1	T1 1 T1	36	3C2
2	1	T1 2 5 4 T1	73	952
3	1	T1 8 3 6 T1	92	834
4	1	T1 7 T1	18	14C

TCTAL DISTANCE = 2228

TRLCK AVAILABILITY AND ALLCCATICN TABLE

TRLCK	10C	9C	8C	5C	3C
AVAILABLE	1	1	2	1	1
ALLOCATEC	1	C	1	1	1

PROBLEM 10

NUMBER OF DEMAND POINTS = 9 NUMBER OF TERMINALS = 1

DEMAND POINT	DEMAND AT THE POINT	DISTANCE TO THE TERMINAL
1	20	125
2	38	186
3	24	49
4	28	174
5	25	53
6	34	68
7	23	38
8	16	152
9	26	92

TRUCKS

CAPACITY	NUMBER
100	1
90	1
70	1
60	1
50	2

DISTANCE MATRIX

(2)	62							
(3)	174	235						
(4)	64	53	223					
(5)	126	178	98	140				
(6)	65	227	117	112	69			
(7)	90	152	87	136	42	32		
(8)	33	44	198	92	159	92	123	
(9)	36	94	141	100	101	34	66	58

FINAL SOLUTION

NO.	TERMINAL	ROUTES	LOAD	DISTANCE
1	1	T1 3 5 T1	49	200
2	1	T1 4 2 1 T1	86	404
3	1	T1 8 9 6 7 T1	99	314

TOTAL DISTANCE = 918

TRUCK AVAILABILITY AND ALLOCATION TABLE

TRUCK	10C	9C	7C	6C	5C
AVAILABLE	1	1	1	1	2
ALLOCATED	1	1	0	0	1

PROBLEM 11

NUMBER OF DEMAND POINTS = 9 NUMBER OF TERMINALS = 1

DEMAND POINT	DEMAND AT THE POINT	DISTANCE TO THE TERMINAL
1	25	244
2	20	107
3	38	224
4	40	154
5	26	108
6	28	113
7	28	290
8	22	284
9	15	207

TRUCKS

CAPACITY	NUMBER
100	1
90	1
80	1
70	2

DISTANCE MATRIX

(2)	139							
(3)	468	331						
(4)	212	152	284					
(5)	309	172	167	121				
(6)	354	217	125	216	114			
(7)	533	396	159	396	294	180		
(8)	134	191	508	330	354	394	559	
(9)	110	132	388	110	221	317	497	228

FINAL SOLUTION

NO.	TERMINAL	ROUTES	LOAD	DISTANCE
1	1	T1 2 8 1 9 T1	82	749
2	1	T1 4 5 T1	66	383
3	1	T1 6 7 3 T1	94	676

TOTAL DISTANCE = 1808

TRUCK AVAILABILITY AND ALLOCATION TABLE

TRUCK	100	90	80	70
AVAILABLE	1	1	1	2
ALLOCATED	1	1	0	1

PROBLEM 12

NUMBER OF DEMAND POINTS =10 NUMBER OF TERMINALS = 1

DEMAND POINT	DEMAND AT THE POINT	DISTANCE TO THE TERMINAL
1	10	8
2	4	5
3	8	3
4	7	2
5	12	12
6	6	8
7	5	3
8	10	8
9	8	7
10	15	7

TRUCKS

CAPACITY	NUMBER
30	10
0	0
0	0

DISTANCE MATRIX

(2)	6								
(3)	5	3							
(4)	10	7	5						
(5)	7	11	10	14					
(6)	6	8	6	10	5				
(7)	8	7	4	4	10	6			
(8)	16	13	11	6	15	14	5		
(9)	9	10	7	8	8	4	4	11	
(10)	13	12	9	7	14	9	6	7	6

FINAL SOLUTION

NO.	TERMINAL	ROUTES					LOAD	DISTANCE
1	1	T1	3	2	1	T1	22	20
2	1	T1	5	6	9	T1	26	28
3	1	T1	7	10	8	T1	30	24
4	1	T1	4	T1			7	4

TOTAL DISTANCE = 76

TRUCK AVAILABILITY AND ALLOCATION TABLE

TRUCK	30	0	0
AVAILABLE	10	0	0
ALLOCATED	4	0	0

PROBLEM 13

NUMBER OF DEMAND POINTS =10 NUMBER OF TERMINALS = 1

DEMAND POINT	DEMAND AT THE POINT	DISTANCE TO THE TERMINAL
1	12	315
2	5	388
3	8	296
4	15	158
5	6	186
6	7	391
7	14	301
8	4	151
9	13	193
10	8	239

TRUCKS

CAPACITY	NUMBER
40	1
30	3
20	10

DISTANCE MATRIX

(2)	695								
(3)	331	502							
(4)	152	478	284						
(5)	172	483	167	121					
(6)	539	272	301	476	367				
(7)	217	569	125	216	114	394			
(8)	376	252	258	251	213	250	323		
(9)	364	290	211	276	192	200	279	70	
(10)	132	572	388	110	221	608	317	368	408

FINAL SCLLTICN

NC.	TERMINAL	ROUTES	LCAD	DISTANCE
1	1	T1 4 1 1C T1	35	681
2	1	T1 5 7 3 T1	28	721
3	1	T1 9 6 2 8 T1	29	1CE8

TCTAL DISTANCE = 247C

TRUCK AVAILABILITY AND ALLCCATICN TABLE

TRUCK	40	3C	2C
AVAILABLE	1	3	1C
ALLOCATED	1	2	C

PROBLEM 14

NUMBER OF DEMAND POINTS =10 NUMBER OF TERMINALS = 1

DEMAND POINT	DEMAND AT THE POINT	DISTANCE TO THE TERMINAL
1	10	151
2	6	321
3	8	258
4	12	251
5	7	213
6	5	250
7	12	323
8	9	170
9	11	70
10	8	390

TRUCKS

CAPACITY	NUMBER
40	2
30	2
20	10

DISTANCE MATRIX

(2)	293								
(3)	296	224							
(4)	158	154	284						
(5)	186	108	167	121					
(6)	391	468	301	476	367				
(7)	301	113	125	216	114	394			
(8)	288	338	172	354	234	130	264		
(9)	193	300	211	276	192	200	279	95	
(10)	511	558	391	574	457	144	484	220	321

FINAL SOLUTION

NO.	TERMINAL	ROUTES						LOAD	DISTANCE
1	1	T1	9	8	10	6	T1	33	779
2	1	T1	3	7	2	5	T1	33	817
3	1	T1	4	1	T1			22	560

TOTAL DISTANCE = 2156

TRUCK AVAILABILITY AND ALLOCATION TABLE

TRUCK	40	30	20
AVAILABLE	2	2	10
ALLOCATED	2	1	0

PROBLEM 15

NUMBER OF DEMAND POINTS =10 NUMBER OF TERMINALS = 1

DEMAND POINT	DEMAND AT THE POINT	DISTANCE TO THE TERMINAL
1	5	230
2	12	194
3	8	244
4	6	107
5	15	265
6	9	154
7	16	108
8	10	326
9	7	113
10	11	207

TRUCKS

CAPACITY	NUMBER
50	1
40	1
30	2
20	10

DISTANCE MATRIX

(2)	232								
(3)	33	265							
(4)	123	168	139						
(5)	190	370	167	202					
(6)	233	325	212	152	160				
(7)	295	301	309	172	277	121			
(8)	253	428	230	252	86	131	248		
(9)	340	262	354	217	375	216	114	347	
(10)	133	309	110	132	61	110	221	119	317

FINAL SCLUTICN

NC.	TERMINAL	ROUTES	LCAD	DISTANCE
1	1	T1 2 1 3 4 T1	31	7C5
2	1	T1 1C 5 8 6 T1	45	63S
3	1	T1 7 S T1	23	335

TCTAL CISTANCE = 167S

TRUCK AVAILABILITY AND ALLCCATION TABLE

TRUCK	5C	4C	3C	2C
AVAILABLE	1	1	2	1C
ALLCCATEC	1	1	1	C

PROBLEM 16

NUMBER OF DEMAND POINTS =10 NUMBER OF TERMINALS = 1

DEMAND POINT	DEMAND AT THE POINT	DISTANCE TO THE TERMINAL
1	20	558
2	15	476
3	8	645
4	16	514
5	7	353
6	9	294
7	17	302
8	10	220
9	14	404
10	5	450

TRUCKS

CAPACITY	NUMBER
50	2
40	2
30	2
20	10

DISTANCE MATRIX

(2)	154								
(3)	805	653							
(4)	400	248	403						
(5)	505	386	325	256					
(6)	366	247	464	210	137				
(7)	252	187	686	375	346	209			
(8)	649	530	425	503	229	283	452		
(9)	783	664	271	543	287	417	634	182	
(10)	895	776	384	733	473	529	698	246	186

FINAL SOLUTION

NO.	TERMINAL	ROUTES	LOAD	DISTANCE
1	1	T1 7 1 T1	37	1112
2	1	T1 2 4 6 T1	40	1228
3	1	T1 5 3 9 10 8 T1	44	1601

TOTAL DISTANCE = 3941

TRUCK AVAILABILITY AND ALLOCATION TABLE

TRUCK	50	40	30	20
AVAILABLE	2	2	2	10
ALLOCATED	1	2	0	0

PROBLEM 17

NUMBER OF DEMAND POINTS =10 NUMBER OF TERMINALS = 1

DEMAND POINT	DEMAND AT THE POINT	DISTANCE TO THE TERMINAL
1	26	468
2	38	293
3	40	343
4	16	315
5	32	296
6	27	158
7	18	186
8	16	301
9	26	451
10	20	239

TRUCKS

CAPACITY	NUMBER
100	1
90	1
80	1
70	2
50	2

DISTANCE MATRIX

(2)	197								
(3)	268	244							
(4)	168	107	139						
(5)	394	224	468	331					
(6)	307	154	212	152	284				
(7)	301	108	309	172	167	121			
(8)	262	113	354	217	125	216	114		
(9)	411	290	533	396	159	396	294	180	
(10)	301	207	110	132	388	110	221	317	497

FINAL SOLUTION

NO.	TERMINAL	ROUTES	LOAD	DISTANCE
1	1	T1 6 3 10 T1	87	719
2	1	T1 4 1 2 T1	80	973
3	1	T1 7 8 9 5 T1	92	935

TOTAL DISTANCE = 2627

TRUCK AVAILABILITY AND ALLOCATION TABLE

TRUCK	100	90	80	70	50
AVAILABLE	1	1	1	2	2
ALLOCATED	1	1	1	0	0

PROBLEM 18

NUMBER OF DEMAND POINTS =10 NUMBER OF TERMINALS = 1

DEMAND POINT	DEMAND AT THE POINT	DISTANCE TO THE TERMINAL
1	29	499
2	30	208
3	22	146
4	24	205
5	30	350
6	28	253
7	17	309
8	15	176
9	18	258
10	25	195

TRUCKS

CAPACITY	NUMBER
100	1
90	1
80	1
70	2
50	2

DISTANCE MATRIX

(2)	704								
(3)	565	272							
(4)	656	145	349						
(5)	816	252	250	372					
(6)	645	336	130	454	170				
(7)	746	290	200	410	70	95			
(8)	421	344	144	381	390	220	321		
(9)	271	471	324	385	570	404	503	182	
(10)	384	403	344	303	548	450	507	246	186

FINAL SOLUTION

NO.	TERMINAL	ROUTES	LOAD	DISTANCE
1	1	T1 1C 1 9 8 T1	87	12C8
2	1	T1 3 6 7 5 T1	97	791
3	1	T1 2 4 T1	54	551

TOTAL DISTANCE = 255C

TRUCK AVAILABILITY AND ALLOCATION TABLE

TRUCK	1C0	90	80	70	5C
AVAILABLE	1	1	1	2	2
ALLOCATED	1	1	C	1	C

PROBLEM 15

NUMBER OF DEMAND POINTS =10 NUMBER OF TERMINALS = 1

DEMAND POINT	DEMAND AT THE POINT	DISTANCE TO THE TERMINAL
1	16	645
2	24	336
3	18	794
4	30	130
5	15	253
6	23	454
7	17	170
8	20	95
9	24	220
10	24	404

TRUCKS

CAPACITY	NUMBER
100	1
90	1
80	2
60	2
40	2

DISTANCE MATRIX

(2)	704								
(3)	245	946							
(4)	565	272	780						
(5)	499	201	741	146					
(6)	656	145	698	394	205				
(7)	815	252	964	250	350	372			
(8)	746	290	895	200	309	410	70		
(9)	425	344	640	144	176	381	390	321	
(10)	271	471	513	324	258	385	570	503	182

FINAL SOLUTION

NO.	TERMINAL	ROUTES	LOAD	DISTANCE
1	1	T1 8 7 T1	37	335
2	1	T1 2 6 5 4 T1	92	962
3	1	T1 1 3 10 9 T1	82	1805

TOTAL DISTANCE = 3102

TRUCK AVAILABILITY AND ALLOCATION TABLE

TRUCK	10C	9C	8C	6C	4C
AVAILABLE	1	1	2	2	2
ALLOCATED	1	1	0	0	1

PROBLEM 20

NUMBER OF DEMAND POINTS =10 NUMBER OF TERMINALS = 1

DEMAND POINT	DEMAND AT THE POINT	DISTANCE TO THE TERMINAL
1	22	58
2	38	33
3	26	115
4	36	52
5	40	55
6	15	25
7	25	78
8	18	150
9	30	32
10	24	43

TRUCKS

CAPACITY	NUMBER
90	1
80	2
70	2
60	2
40	2

DISTANCE MATRIX

(2)	64								
(3)	173	128							
(4)	110	74	63						
(5)	113	68	61	29					
(6)	83	38	90	36	30				
(7)	131	110	50	55	84	89			
(8)	208	163	40	98	96	125	90		
(9)	35	65	146	84	87	57	96	182	
(10)	15	49	158	95	98	68	119	193	24

FINAL SCLLTICN

NC.	TERMINAL	ROUTES	LCAD	DISTANCE
1	1	T1 9 1 1C T1	76	125
2	1	T1 6 4 7 T1	76	194
3	1	T1 3 8 5 T1	64	3C6
4	1	T1 2 T1	38	66

TCTAL CISTANCE = 691

TRUCK AVAILABILITY ANC ALLCCATION TABLE

TRLCK	9C	80	70	6C	4C
AVAILABLE	1	2	2	2	2
ALLCCATED	1	2	C	C	1

PROBLEM 21

NUMBER OF DEMAND POINTS =14 NUMBER OF TERMINALS = 1

DEMAND POINT	DEMAND AT THE POINT	DISTANCE TO THE TERMINAL
1	36	102
2	28	143
3	18	47
4	24	88
5	36	71
6	15	116
7	30	30
8	22	32
9	18	44
10	27	18
11	18	103
12	34	63
13	40	118
14	17	142

TRUCKS

CAPACITY	NUMBER
100	3
90	2
70	1
60	1
50	2

DISTANCE MATRIX

(2)	205											
(3)	149	136										
(4)	167	55	90									
(5)	84	183	118	146								
(6)	195	29	107	28	173							
(7)	121	114	35	59	93	86						
(8)	70	141	79	104	42	132	53					
(9)	100	111	59	74	72	104	23	30				
(10)	117	161	36	106	88	134	47	49	63			
(11)	20	213	149	176	66	204	125	73	103	118		
(12)	59	176	110	138	26	166	85	34	64	82	41	
(13)	139	74	126	37	146	65	91	105	75	136	146	120
(14)	141	75	150	78	170	76	115	129	99	160	149	144	41

FINAL SOLUTION

NO.	TERMINAL	ROUTES	LOAD	DISTANCE
1	1	T1 8 9 T1	40	106
2	1	T1 7 4 6 2 T1	97	289
3	1	T1 14 13 5 T1	93	400
4	1	T1 11 1 12 T1	88	245
5	1	T1 3 10 T1	45	101

TOTAL DISTANCE = 1141

TRUCK AVAILABILITY AND ALLOCATION TABLE

TRUCK	100	90	70	60	50
AVAILABLE	3	2	1	1	2
ALLOCATED	2	1	0	0	2

PROBLEM 22

NUMBER OF DEMAND POINTS = 25 NUMBER OF TERMINALS = 1

DEMAND POINT	DEMAND AT THE POINT	DISTANCE TO THE TERMINAL
1	18	656
2	26	468
3	38	293
4	40	343
5	27	785
6	16	315
7	29	936
8	30	338
9	32	296
10	27	158
11	18	186
12	38	764
13	22	391
14	16	301
15	35	544
16	26	451
17	18	944
18	24	476
19	30	151
20	28	288
21	17	193
22	15	511
23	20	239
24	18	693
25	25	697

TRUCKS

CAPACITY	NUMBER
100	3
90	4
80	2
70	5
60	1
50	6
40	2
30	8
20	10

DISTANCE MATRIX

(2)	574																								
(3)	476	197																							
(4)	716	268	244																						
(5)	367	388	517	655																					
(6)	579	168	107	139	557																				
(7)	653	1075	943	1187	1020	1050																			
(8)	808	785	571	679	1064	695	704																		
(9)	364	394	224	468	586	331	734	502																	
(10)	582	307	154	212	671	152	999	478	284																
(11)	480	301	108	309	601	172	882	483	167	121															
(12)	248	786	662	906	615	769	403	828	468	748	631														
(13)	567	663	468	715	870	539	565	272	301	476	367	556													
(14)	366	262	113	354	499	217	833	569	125	216	114	552	394												
(15)	247	611	502	723	581	586	464	608	273	553	436	210	336	389											
(16)	187	411	290	533	457	396	686	638	159	396	294	375	423	180	209										
(17)	351	925	820	1063	697	926	498	989	642	926	805	181	730	710	397	530									
(18)	911	885	691	817	1184	748	656	145	622	616	583	866	349	689	664	756	1026								
(19)	611	513	321	475	812	376	815	252	258	251	213	669	250	323	449	406	850	372							
(20)	476	533	338	546	749	409	645	336	172	354	234	514	130	264	294	302	695	454	170						
(21)	541	497	300	515	778	364	746	290	211	276	192	598	200	279	383	355	779	410	70	95					
(22)	530	753	558	766	864	629	425	344	391	574	457	503	144	484	283	452	645	381	390	220	321				
(23)	683	301	207	110	689	132	1154	572	388	110	221	869	608	317	686	497	1027	710	368	509	408	729			
(24)	664	935	740	948	998	811	271	471	573	756	641	543	324	668	417	634	689	385	570	404	503	182	911		
(25)	776	981	788	1022	1100	852	384	403	621	783	682	733	344	714	529	698	833	303	548	450	507	246	915	186	

FINAL SOLUTION

NO.	TERMINAL	ROUTES						LOAD	DISTANCE
1	1	T1	10	23	4	T1		87	721
2	1	T1	6	2	3	11	T1	98	974
3	1	T1	14	5	1	16	T1	87	1805
4	1	T1	9	15	13	T1		89	1296
5	1	T1	20	21	19	T1		75	604
6	1	T1	8	24	25	18	T1	97	1774
7	1	T1	22	7	17	12	T1	100	2379

TOTAL DISTANCE = 9553

TRUCK AVAILABILITY AND ALLOCATION TABLE

TRUCK	100	90	80	70	60	50	40	30	20
AVAILABLE	3	4	2	5	1	6	2	8	10
ALLOCATED	3	3	1	0	0	0	0	0	0

PROBLEM 23

NUMBER OF DEMAND POINTS = 25 NUMBER OF TERMINALS = 1

DEMAND POINT	DEMAND AT THE POINT	DISTANCE TO THE TERMINAL
1	16	476
2	25	546
3	36	749
4	20	409
5	16	645
6	24	336
7	38	172
8	40	354
9	18	794
10	26	234
11	25	514
12	30	130
13	15	253
14	28	264
15	19	294
16	37	572
17	28	302
18	18	695
19	23	454
20	17	170
21	22	591
22	20	95
23	24	220
24	15	509
25	24	404

TRUCKS

CAPACITY	NUMBER
100	4
90	4
80	2
70	2
60	4
50	2
40	2
30	1
20	10

DISTANCE MATRIX

(2)	716																								
(3)	367	655																							
(4)	579	139	557																						
(5)	653	1187	1020	1050																					
(6)	808	679	1064	695	704																				
(7)	364	468	586	331	734	502																			
(8)	582	212	671	152	999	478	284																		
(9)	662	1284	1029	1147	245	946	834	1114																	
(10)	480	309	601	172	882	483	167	121	997																
(11)	248	906	615	769	403	828	468	748	429	631															
(12)	567	715	870	539	565	272	301	476	780	367	556														
(13)	706	824	1001	656	499	201	423	585	741	484	661	146													
(14)	366	354	499	217	833	569	125	216	930	114	552	394	516												
(15)	247	723	581	586	464	608	273	553	561	436	210	336	459	389											
(16)	94	793	378	656	650	900	456	674	659	572	245	659	798	458	339										
(17)	187	533	457	396	686	638	159	396	783	294	375	423	554	180	209	297									
(18)	351	1063	697	926	498	989	642	926	359	805	181	730	821	710	397	324	530								
(19)	911	817	1184	748	656	145	622	616	898	583	866	349	205	689	664	1003	756	1026							
(20)	611	475	812	376	815	252	258	251	964	213	669	250	350	323	449	703	406	850	372						
(21)	742	134	568	191	1232	804	508	330	1319	354	934	721	937	394	758	819	559	1093	942	588					
(22)	541	515	778	364	746	290	211	276	895	192	598	200	309	279	383	633	355	779	410	70	628				
(23)	530	766	864	629	425	344	391	574	640	457	503	144	176	484	283	622	452	645	381	390	811	321			
(24)	683	110	689	132	1154	572	388	110	1247	221	869	608	717	317	686	775	497	1027	710	368	228	408	729		
(25)	664	948	998	811	271	471	573	756	513	641	543	324	258	668	417	756	634	689	385	570	995	503	182	911	

FINAL SCLLTICN

NC.	TERMINAL	ROUTES	LCAD	DISTANCE
1	1	T1 12 25 23 T1	78	856
2	1	T1 13 19 6 T1	62	939
3	1	T1 5 9 18 11 15 T1	96	1934
4	1	T1 1 16 3 T1	89	1697
5	1	T1 4 21 2 24 T1	82	1353
6	1	T1 10 8 14 T1	94	835
7	1	T1 17 7 T1	66	633
8	1	T1 20 22 T1	37	335

TCTAL CISTANCE = 8582

TRUCK AVAILABILITY AND ALLCCATICN TABLE

TRUCK	100	90	80	70	60	50	40	30	20
AVAILABLE	4	4	2	2	4	2	2	1	10
ALLOCATED	2	2	1	2	0	0	1	0	0

PROBLEM 24

NUMBER OF DEMAND POINTS = 25 NUMBER OF TERMINALS = 1

DEMAND POINT	DEMAND AT THE POINT	DISTANCE TO THE TERMINAL
1	18	91
2	22	58
3	35	222
4	20	109
5	16	136
6	38	33
7	34	98
8	24	157
9	26	115
10	35	52
11	15	119
12	27	113
13	17	181
14	23	220
15	36	132
16	40	55
17	15	25
18	16	188
19	28	156
20	32	136
21	22	161
22	25	78
23	18	150
24	30	32
25	24	43

TRUCKS

CAPACITY	NUMBER
100	2
90	4
80	3
70	2
60	2
50	2
40	4
30	3
20	10

DISTANCE MATRIX

(2)	149																							
(3)	140	258																						
(4)	184	127	149																					
(5)	216	89	198	49																				
(6)	94	64	194	90	122																			
(7)	99	134	124	88	133	70																		
(8)	212	177	142	50	98	124	117																	
(9)	101	173	239	218	250	128	146	252																
(10)	89	110	227	161	188	74	109	198	63															
(11)	195	88	188	40	21	101	112	89	229	171														
(12)	24	154	118	167	212	90	79	192	123	111	191													
(13)	118	217	53	96	145	153	83	106	217	192	136	96												
(14)	146	256	26	132	181	192	122	116	245	231	172	124	40											
(15)	43	170	100	162	207	106	76	186	142	130	186	19	81	109										
(16)	61	113	199	158	190	68	85	192	61	29	169	83	168	205	102									
(17)	66	83	197	128	160	38	73	162	90	36	139	88	156	195	107	30								
(18)	150	224	58	100	149	160	90	84	236	199	140	128	31	32	114	175	163							
(19)	135	192	78	71	120	128	58	81	204	167	111	115	25	65	105	143	131	33						
(20)	212	118	177	36	39	118	124	64	246	188	30	203	124	160	198	186	156	128	99					
(21)	237	156	172	60	74	143	147	30	271	213	65	222	136	146	216	211	181	114	111	35				
(22)	144	131	282	185	212	110	162	234	50	55	195	166	245	284	185	84	89	252	220	212	237			
(23)	93	208	231	253	285	163	181	287	40	98	264	115	209	237	134	96	125	241	220	281	306	90		
(24)	123	35	254	136	117	65	130	186	146	84	111	145	213	252	164	87	57	220	188	141	176	96	182	
(25)	134	15	243	112	93	49	119	162	158	95	87	139	202	241	155	98	68	209	177	117	152	119	193	24

FINAL SOLUTION

NO.	TERMINAL	ROUTES	LOAD	DISTANCE
1	1	T1 17 16 10 T1	90	136
2	1	T1 22 9 23 T1	69	318
3	1	T1 1 12 15 T1	81	266
4	1	T1 13 3 14 18 T1	91	480
5	1	T1 7 19 T1	62	302
6	1	T1 6 T1	38	66
7	1	T1 4 8 21 20 T1	98	360
8	1	T1 5 11 2 25 T1	77	303
9	1	T1 24 T1	30	64

TOTAL DISTANCE = 2295

TRUCK AVAILABILITY AND ALLOCATION TABLE

TRUCK	100	90	80	70	60	50	40	30	20
AVAILABLE	2	4	3	2	2	2	4	3	10
ALLOCATED	2	2	1	2	0	0	1	1	0

PROBLEM 25

NUMBER OF DEMAND POINTS = 25 NUMBER OF TERMINALS = 1

DEMAND POINT	DEMAND AT THE POINT	DISTANCE TO THE TERMINAL
1	20	125
2	16	268
3	38	186
4	24	49
5	36	116
6	28	233
7	18	163
8	24	195
9	36	50
10	22	203
11	28	174
12	15	223
13	30	140
14	22	91
15	18	121
16	25	53
17	27	132
18	34	68
19	23	38
20	16	152
21	18	97
22	26	92
23	34	69
24	40	189
25	17	212

TRUCKS

CAPACITY	NUMBER
100	2
90	3
80	4
70	2
60	1
50	2
40	2
30	1
20	10

DISTANCE MATRIX

(2)	171																								
(3)	62	155																							
(4)	174	310	235																						
(5)	166	240	196	89																					
(6)	217	81	227	257	205																				
(7)	81	107	56	210	149	136																			
(8)	168	73	184	237	167	55	90																		
(9)	95	219	146	91	84	183	118	146																	
(10)	91	88	68	249	188	170	40	124	158																
(11)	64	207	53	223	232	279	144	232	160	120															
(12)	189	60	204	265	195	29	107	28	173	139	249	00													
(13)	111	132	126	185	121	114	35	59	93	75	173	86													
(14)	109	177	127	133	70	141	79	104	42	119	174	132	53												
(15)	125	147	143	167	100	111	59	74	72	99	189	102	23	30											
(16)	126	287	178	98	169	273	180	225	90	217	140	253	166	132	164										
(17)	64	143	84	178	117	161	36	106	88	76	128	134	47	49	63	146									
(18)	65	220	227	117	144	212	113	157	60	153	112	185	89	94	114	69	79								
(19)	90	251	152	87	149	244	145	189	64	181	136	217	131	106	136	42	110	32							
(20)	33	137	44	198	152	188	52	140	110	63	92	160	82	83	99	159	40	92	123						
(21)	147	250	172	70	20	213	149	176	66	189	211	204	125	73	103	150	118	126	130	154					
(22)	36	189	94	141	126	198	82	143	59	121	100	111	84	79	99	101	48	34	66	58	118				
(23)	105	212	149	104	59	176	110	138	26	150	170	166	85	34	64	116	82	86	90	113	41	77			
(24)	200	110	216	213	139	74	126	37	146	166	264	65	91	105	75	236	136	187	210	172	146	172	120		
(25)	224	136	240	215	141	75	150	78	170	190	288	76	115	129	99	261	160	211	235	196	149	196	144	41	

FINAL SOLUTION

NO.	TERMINAL	ROUTES	LOAD	DISTANCE
1	1	T1 4 5 21 T1	78	255
2	1	T1 23 9 T1	70	145
3	1	T1 6 12 2 10 7 T1	99	613
4	1	T1 25 24 8 15 T1	99	485
5	1	T1 14 13 17 T1	79	323
6	1	T1 20 3 11 T1	82	423
7	1	T1 22 1 18 T1	80	261
8	1	T1 19 16 T1	48	133

TOTAL DISTANCE = 2638

TRUCK AVAILABILITY AND ALLOCATION TABLE

TRUCK	100	90	80	70	60	50	40	30	20
AVAILABLE	2	3	4	2	1	2	2	1	10
ALLOCATED	2	1	3	1	0	1	0	0	0

PROBLEM 26

NUMBER OF DEMAND POINTS = 25 NUMBER OF TERMINALS = 1

DEMAND POINT	DEMAND AT THE POINT	DISTANCE TO THE TERMINAL
1	25	252
2	18	187
3	36	451
4	24	290
5	15	533
6	17	457
7	28	686
8	23	638
9	16	159
10	34	396
11	22	783
12	18	294
13	30	423
14	27	554
15	16	346
16	24	180
17	17	209
18	28	297
19	36	530
20	23	756
21	22	302
22	18	355
23	40	452
24	25	497
25	19	698

TRUCKS

CAPACITY	NUMBER
100	4
90	3
80	4
70	2
60	2
50	1
40	2
30	1
20	10

DISTANCE MATRIX

(2)	154																							
(3)	730	656																						
(4)	461	476	293																					
(5)	686	716	343	244																				
(6)	215	367	785	517	655																			
(7)	805	653	936	943	1187	1020																		
(8)	894	808	338	571	679	1064	704																	
(9)	415	364	296	224	468	586	734	502																
(10)	610	582	158	154	212	671	995	478	284															
(11)	814	662	1085	1040	1284	1029	245	946	834	1114														
(12)	508	480	186	109	309	601	882	483	167	121	997													
(13)	668	567	351	468	715	870	565	272	301	476	780	367												
(14)	810	706	499	590	824	1001	495	201	423	585	741	484	146											
(15)	505	386	644	618	862	720	325	573	409	679	445	562	354	405										
(16)	432	366	301	113	354	499	833	569	125	216	930	114	394	516	508									
(17)	366	247	544	502	723	581	464	608	273	553	561	436	336	459	137	389								
(18)	173	94	748	568	793	378	650	500	456	674	659	572	659	798	478	458	339							
(19)	493	351	544	820	1063	697	498	589	642	926	359	805	730	821	416	710	397	324						
(20)	1012	911	476	691	817	1184	656	145	622	616	898	583	349	205	610	689	664	1003	1026					
(21)	558	476	288	338	546	749	645	336	172	354	794	234	130	253	353	264	294	572	695	454				
(22)	611	541	193	300	515	778	746	290	211	276	895	192	200	309	454	279	383	633	779	410	95			
(23)	649	530	511	558	766	864	425	344	391	574	640	457	144	176	229	484	283	622	645	381	220	321		
(24)	668	683	239	207	110	689	1154	572	388	110	1247	221	608	717	783	317	686	775	1027	710	509	408	729	
(25)	895	776	697	788	1022	1100	384	403	621	783	612	682	344	195	473	714	529	868	833	303	450	507	246	915

FINAL SOLUTION

NO.	TERMINAL	ROUTES	LOAD	DISTANCE
1	1	T1 2 18 1 6 T1	88	1126
2	1	T1 17 15 9 T1	49	914
3	1	T1 21 13 23 T1	88	1028
4	1	T1 8 20 25 14 T1	92	1835
5	1	T1 7 11 19 T1	86	1820
6	1	T1 16 12 3 22 T1	96	1028
7	1	T1 4 5 24 10 T1	98	1150

TOTAL DISTANCE = 8901

TRUCK AVAILABILITY AND ALLOCATION TABLE

TRUCK	100	90	80	70	60	50	40	30	20
AVAILABLE	4	3	4	2	2	1	2	1	10
ALLOCATED	3	3	0	0	0	1	0	0	0

APPENDIX C

MULTI-TERMINAL PROBLEMS

PROBLEM 1

NUMBER OF DEMAND POINTS =10 NUMBER OF TERMINALS = 3

DEMAND POINT	DEMAND AT THE POINT	DISTANCE TO THE TERMINAL NO. (1)	DISTANCE TO THE TERMINAL NO. (2)	DISTANCE TO THE TERMINAL NO. (3
1	10	3	8	12
2	4	6	5	12
3	8	5	3	8
4	7	9	2	6
5	12	5	12	14
6	6	3	8	9
7	5	6	3	4
8	10	15	8	6
9	8	7	7	5
10	15	12	7	2

TRUCKS

CAPACITY	NUMBER
30	10

DISTANCE MATRIX

(2)	6								
(3)	5	3							
(4)	10	7	5						
(5)	7	11	10	14					
(6)	6	8	6	10	5				
(7)	8	7	4	4	10	6			
(8)	16	13	11	6	19	14	9		
(9)	9	10	7	8	8	4	4	11	
(10)	13	12	9	7	14	9	6	7	6

FINAL SOLUTION

NO.	TERMINAL	ROUTES	LOAD	DISTANCE
1	1	T1 6 9 5 T1	26	20
2	1	T1 1 T1	10	6
1	2	T2 2 3 7 4 T2	24	18
1	3	T3 8 10 T3	25	15

TOTAL DISTANCE = 59

TRUCK AVAILABILITY AND ALLOCATION TABLE

TRUCK	30
AVAILABLE	10
ALLOCATED	4

PROBLEM 2

NUMBER OF DEMAND POINTS =10 NUMBER OF TERMINALS = 3

DEMAND POINT	DEMAND AT THE POINT	DISTANCE TO THE TERMINAL NO. (1)	DISTANCE TO THE TERMINAL NO. (2)	DISTANCE TO THE TERMINAL NO. (3
1	12	315	409	107
2	5	388	336	571
3	8	296	172	224
4	15	158	354	154
5	6	186	234	108
6	7	391	130	468
7	14	301	264	113
8	4	151	170	321
9	13	193	95	300
10	8	239	509	207

TRUCKS

CAPACITY	NUMBER
40	1
30	3
20	10

DISTANCE MATRIX

(2)	695								
(3)	331	502							
(4)	152	478	284						
(5)	172	483	167	121					
(6)	539	272	301	476	367				
(7)	217	569	125	216	114	394			
(8)	376	252	258	251	213	250	323		
(9)	364	290	211	276	192	200	279	70	
(10)	132	572	388	110	221	608	317	368	408

FINAL SOLUTION

NO.	TERMINAL	ROUTES						LOAD	DISTANCE
1	2	T2	9	8	2	6	T2	29	819
1	3	T3	7	3	5	T3		28	513
2	3	T3	4	10	1	T3		35	503

TOTAL DISTANCE = 1835

TRUCK AVAILABILITY AND ALLOCATION TABLE

TRUCK	40	30	20
AVAILABLE	1	3	10
ALLOCATED	1	2	0

PROBLEM 3

NUMBER OF DEMAND POINTS =10 NUMBER OF TERMINALS = 3

DEMAND POINT	DEMAND AT THE POINT	DISTANCE TO THE TERMINAL NO. (1)	DISTANCE TO THE TERMINAL NO. (2)	DISTANCE TO THE TERMINAL NO. (3
1	10	499	151	315
2	6	590	321	107
3	8	423	258	331
4	12	585	251	152
5	7	484	213	172
6	5	146	250	539
7	12	516	323	217
8	9	253	170	409
9	11	309	70	364
10	8	176	390	629

TRUCKS

CAPACITY	NUMBER
40	2
30	2
20	10

DISTANCE MATRIX

(2)	293								
(3)	296	224							
(4)	158	154	284						
(5)	186	108	167	121					
(6)	391	468	301	476	367				
(7)	301	113	125	216	114	394			
(8)	288	338	172	354	234	130	264		
(9)	193	300	211	276	192	200	279	95	
(10)	511	558	391	574	457	144	484	220	321

FINAL SOLUTION

NO.	TERMINAL	ROUTES	LOAD	DISTANCE
1	1	T1 6 10 T1	13	466
1	2	T2 1 3 8 9 T2	38	784
1	3	T3 2 7 5 4 T3	37	607

TOTAL DISTANCE = 1857

TRUCK AVAILABILITY AND ALLOCATION TABLE

TRUCK	40	30	20
AVAILABLE	2	2	10
ALLOCATED	2	0	1

PROBLEM 4

NUMBER OF DEMAND POINTS =10 NUMBER OF TERMINALS = 3

DEMAND POINT	DEMAND AT THE POINT	DISTANCE TO THE TERMINAL NO. (1)	DISTANCE TO THE TERMINAL NO. (2)	DISTANCE TO THE TERMINAL NO. (3
1	5	366	230	105
2	12	483	194	219
3	8	343	244	134
4	6	315	107	182
5	15	271	265	285
6	9	158	154	330
7	16	186	108	354
8	10	215	326	348
9	7	301	113	394
10	11	239	207	228

TRUCKS

CAPACITY	NUMBER
50	1
40	1
30	2
20	5

DISTANCE MATRIX

(2)	232								
(3)	33	265							
(4)	123	168	139						
(5)	190	370	167	202					
(6)	233	325	212	152	160				
(7)	295	301	309	172	277	121			
(8)	253	428	230	252	86	131	248		
(9)	340	262	354	217	375	216	114	347	
(10)	133	309	110	132	61	110	221	119	317

FINAL SOLUTION

NO.	TERMINAL	ROUTES						LOAD	DISTANCE
1	2	T2	6	8	5	10	T2	45	639
2	2	T2	7	9	T2			23	335
1	3	T3	1	3	4	2	T3	31	664

TOTAL DISTANCE = 1638

TRUCK AVAILABILITY AND ALLOCATION TABLE

TRUCK	50	40	30	20
AVAILABLE	1	1	2	5
ALLOCATED	1	1	1	0

PROBLEM 5

NUMBER OF DEMAND POINTS =10 NUMBER OF TERMINALS = 3

DEMAND POINT	DEMAND AT THE POINT	DISTANCE TO THE TERMINAL NO. (1)	DISTANCE TO THE TERMINAL NO. (2)	DISTANCE TO THE TERMINAL NO. (3
1	20	810	558	432
2	15	706	476	366
3	8	499	645	833
4	16	661	514	552
5	7	405	353	508
6	9	459	294	389
7	17	554	302	180
8	10	176	220	484
9	14	258	404	668
10	5	195	450	714

TRUCKS

CAPACITY	NUMBER
50	2
40	2
30	2
20	5

DISTANCE MATRIX

(2)	154								
(3)	805	653							
(4)	400	248	403						
(5)	505	386	325	256					
(6)	366	247	464	210	137				
(7)	252	187	686	375	346	209			
(8)	649	530	425	503	229	283	452		
(9)	783	664	271	543	287	417	634	182	
(10)	895	776	384	733	473	529	698	246	186

FINAL SOLUTION

NO.	TERMINAL	ROUTES	LOAD	DISTANCE
1	1	T1 8 9 3 10 T1	37	1208
1	3	T3 7 1 T3	37	864
2	3	T3 6 5 4 2 T3	47	1396

TOTAL DISTANCE = 3468

TRUCK AVAILABILITY AND ALLOCATION TABLE

TRUCK	50	40	30	20
AVAILABLE	2	2	2	5
ALLOCATED	1	2	0	0

PROBLEM 6

NUMBER OF DEMAND POINTS =25 NUMBER OF TERMINALS = 5

DEMAND POINT	DEMAND AT THE POINT	DISTANCE TO THE TERMINAL NO. (1)	DISTANCE TO THE TERMINAL NO. (2)	DISTANCE TO THE TERMINAL NO. (3)	DISTANCE TO THE TERMINAL NO. (4)	DISTANCE TO THE TERMINAL NO. (5)
1	18	706	656	884	386	154
2	26	785	468	510	750	497
3	38	590	293	578	618	461
4	40	824	343	426	862	686
5	27	1001	785	605	720	215
6	16	656	315	476	725	549
7	29	499	936	1526	325	805
8	30	201	338	1092	573	894
9	32	423	296	807	409	415
10	27	585	158	631	679	610
11	18	484	186	648	562	508
12	38	661	764	1132	256	400
13	22	146	391	1015	354	668
14	16	516	301	688	508	432
15	35	459	544	1057	137	366
16	26	554	451	868	346	252
17	18	821	944	1235	416	493
18	24	205	476	1230	610	1012
19	30	350	151	889	523	662
20	28	253	288	885	353	558
21	17	309	193	929	454	611
22	15	176	511	1105	229	649
23	20	717	239	529	783	668
24	18	258	693	1289	287	783
25	25	195	697	1436	1453	895

TRUCKS

CAPACITY	NUMBER
100	4
90	4
80	2
70	2
60	4
50	2
40	2
30	1
20	10

DISTANCE MATRIX

(2)	574																								
(3)	476	197																							
(4)	716	268	244																						
(5)	367	388	517	655																					
(6)	579	168	107	139	557																				
(7)	653	1075	943	1187	1020	1050																			
(8)	808	785	571	679	1064	695	704																		
(9)	364	394	224	468	586	331	734	502																	
(10)	582	307	154	212	671	152	995	478	284																
(11)	480	301	108	309	601	172	882	483	167	121															
(12)	248	786	662	906	615	769	403	828	468	748	631														
(13)	567	663	468	715	870	539	565	272	301	476	367	556													
(14)	366	262	113	354	499	217	833	569	125	216	114	552	394												
(15)	247	611	502	723	581	586	464	608	273	553	436	210	336	389											
(16)	187	411	290	533	457	396	686	638	159	396	294	375	423	180	209										
(17)	351	925	820	1063	697	926	498	989	642	926	805	181	730	710	397	530									
(18)	911	885	691	817	1184	748	656	145	622	616	583	866	349	689	664	756	1026								
(19)	611	513	321	475	812	376	815	252	258	251	213	669	250	323	449	406	850	372							
(20)	476	533	338	546	749	409	645	336	172	354	234	514	130	264	294	302	695	454	170						
(21)	541	497	300	515	778	364	746	290	211	276	192	598	200	279	383	355	779	410	70	95					
(22)	530	753	558	766	864	629	425	344	391	574	457	503	144	484	283	452	645	381	390	220	321				
(23)	683	301	207	110	689	132	1154	572	388	110	221	869	608	317	686	497	1027	710	368	509	408	729			
(24)	664	935	740	948	998	811	271	471	573	756	641	543	324	668	417	634	689	385	570	404	503	182	911		
(25)	776	981	788	1022	1100	852	384	403	621	783	682	733	344	714	529	698	833	303	548	450	507	246	915	186	

FINAL SOLUTION

NO.	TERMINAL	ROUTES	LOAD	DISTANCE
1	1	T1 25 7 24 22 T1	87	1208
2	1	T1 13 20 21 19 T1	97	791
3	1	T1 8 18 T1	54	551
1	2	T2 10 4 23 T2	87	719
2	2	T2 6 2 3 T2	80	973
3	2	T2 11 14 16 9 T2	92	935
1	4	T4 17 12 15 T4	91	944
1	5	T5 1 5 T5	45	736

TOTAL DISTANCE = 6857

TRUCK AVAILABILITY AND ALLOCATION TABLE

TRUCK	100	90	80	70	60	50	40	30	20
AVAILABLE	4	4	2	2	4	2	2	1	10
ALLOCATED	3	2	1	0	1	1	0	0	0

PROBLEM 7

NUMBER OF DEMAND POINTS =25 NUMBER OF TERMINALS = 5

DEMAND POINT	DEMAND AT THE POINT	DISTANCE TO THE TERMINAL NO. (1)	DISTANCE TO THE TERMINAL NO. (2)	DISTANCE TO THE TERMINAL NO. (3)	DISTANCE TO THE TERMINAL NO. (4)	DISTANCE TO THE TERMINAL NO. (5)
1	16	386	476	656	476	154
2	25	862	546	343	244	686
3	36	720	749	785	517	215
4	20	725	409	315	107	549
5	16	325	645	936	943	805
6	24	573	336	338	571	894
7	38	409	172	296	224	415
8	40	679	354	158	154	610
9	18	445	794	1085	1040	814
10	26	562	234	186	108	508
11	25	256	514	764	662	400
12	30	354	130	391	468	668
13	15	405	253	499	590	810
14	28	508	264	301	113	432
15	19	137	294	544	502	366
16	37	478	572	748	568	173
17	28	346	302	451	290	252
18	18	416	695	944	820	493
19	23	610	454	476	691	1012
20	17	523	170	151	321	662
21	22	897	591	461	284	697
22	20	454	95	193	300	611
23	24	229	220	511	558	649
24	15	783	509	239	207	668
25	24	287	404	693	740	783

TRUCKS

CAPACITY	NUMBER
100	4
90	4
80	2
70	2
60	4
50	2
40	2
30	1
20	99

DISTANCE MATRIX

(2)	716																							
(3)	367	655																						
(4)	579	139	557																					
(5)	653	1187	1020	1050																				
(6)	808	679	1064	695	704																			
(7)	364	468	586	331	734	502																		
(8)	582	212	671	152	999	478	284																	
(9)	662	1284	1029	1147	245	946	834	1114																
(10)	480	309	601	172	882	483	167	121	997															
(11)	248	906	615	769	403	828	468	748	429	631														
(12)	567	715	870	539	565	272	301	476	780	367	556													
(13)	706	824	1001	656	499	201	423	585	741	484	661	146												
(14)	366	354	459	217	833	569	125	216	930	114	552	394	516											
(15)	247	723	561	586	464	608	273	553	561	436	210	336	459	389										
(16)	94	793	378	656	650	900	456	674	659	572	245	659	798	458	339									
(17)	187	533	457	396	686	638	159	396	783	294	375	423	554	180	209	297								
(18)	351	1063	697	926	498	989	642	926	359	805	181	730	821	710	397	324	530							
(19)	911	817	1184	748	656	145	622	616	898	583	866	349	205	689	664	1003	756	1026						
(20)	611	475	812	376	815	252	258	251	964	213	669	250	350	323	449	703	406	850	372					
(21)	742	134	568	191	1232	804	508	330	1319	354	934	721	937	394	758	819	559	1093	942	588				
(22)	541	515	778	364	746	290	211	276	895	192	598	200	309	279	383	633	355	779	410	70	628			
(23)	530	766	864	629	425	344	391	574	640	457	503	144	176	484	283	622	452	645	381	390	811	321		
(24)	683	110	689	132	1154	572	388	110	1247	221	869	608	717	317	686	775	497	1027	710	368	228	408	729	
(25)	664	948	998	811	271	471	573	756	513	641	543	324	258	668	417	756	634	689	385	570	995	503	182	911

FINAL SOLUTION

NO.	TERMINAL	ROUTES	LOAD	DISTANCE
1	1	T1 15 11 18 T1	62	944
1	2	T2 22 20 T2	37	335
2	2	T2 6 19 13 12 T2	92	962
3	2	T2 5 9 25 23 T2	82	1805
1	4	T4 4 21 2 24 T4	82	749
2	4	T4 8 10 T4	66	383
3	4	T4 14 17 7 T4	94	676
1	5	T5 3 16 1 T5	89	841

TOTAL DISTANCE = 6695

TRUCK AVAILABILITY AND ALLOCATION TABLE

TRUCK	100	90	80	70	60	50	40	30	20
AVAILABLE	4	4	2	2	4	2	2	1	99
ALLOCATED	2	3	0	2	0	0	1	0	0

PROBLEM 8

NUMBER OF DEMAND POINTS =25 NUMBER OF TERMINALS = 5

DEMAND POINT	DEMAND AT THE POINT	DISTANCE TO THE TERMINAL NO. (1)	DISTANCE TO THE TERMINAL NO. (2)	DISTANCE TO THE TERMINAL NO. (3)	DISTANCE TO THE TERMINAL NO. (4)	DISTANCE TO THE TERMINAL NO. (5)
1	18	175	152	26	88	91
2	22	142	246	248	204	58
3	35	131	51	164	54	222
4	20	18	122	193	110	109
5	16	67	171	225	155	136
6	38	83	182	103	140	33
7	34	76	112	115	70	98
8	24	41	94	227	134	157
9	26	211	251	77	187	115
10	36	157	221	63	175	52
11	15	58	162	204	134	119
12	27	155	130	48	66	113
13	17	78	40	142	32	181
14	23	114	25	170	60	220
15	36	150	115	67	54	132
16	40	151	197	35	147	55
17	15	121	185	65	143	25
18	16	82	22	174	64	188
19	28	53	55	151	53	156
20	32	54	150	221	146	136
21	22	71	124	246	164	161
22	25	193	274	118	230	78
23	18	246	243	71	179	150
24	30	148	242	122	200	32
25	24	127	231	133	168	43

TRUCKS

CAPACITY	NUMBER
100	2
90	4
80	3
70	2
60	2
50	2
40	4
30	3
20	99

DISTANCE MATRIX

(2)	149																								
(3)	140	258																							
(4)	184	127	149																						
(5)	216	89	198	49																					
(6)	94	64	194	90	122																				
(7)	99	134	124	88	133	70																			
(8)	212	177	142	50	98	124	117																		
(9)	101	173	239	218	250	128	146	252																	
(10)	89	110	227	161	188	74	109	198	63																
(11)	195	88	188	40	21	101	112	89	229	171															
(12)	24	154	118	167	212	90	79	192	123	111	191														
(13)	118	217	53	96	145	153	83	106	217	192	136	96													
(14)	146	256	26	132	181	192	122	116	245	231	172	124	40												
(15)	43	170	100	162	207	106	76	186	142	130	186	19	81	109											
(16)	61	113	199	158	190	68	85	192	61	29	169	82	168	205	102										
(17)	66	83	197	128	160	38	73	162	90	36	139	88	156	195	107	30									
(18)	150	224	58	100	149	160	90	84	236	199	140	128	31	32	114	175	163								
(19)	135	192	78	71	120	128	58	81	204	167	111	115	25	65	105	143	131	33							
(20)	212	118	177	36	39	118	124	64	246	188	30	203	124	160	198	186	156	128	99						
(21)	237	156	172	60	74	143	147	30	271	213	65	222	136	146	216	211	181	114	111	35					
(22)	144	131	282	185	212	110	162	234	50	55	195	166	245	284	185	84	89	252	220	212	237				
(23)	93	208	231	253	285	163	181	287	40	98	264	115	209	237	134	96	125	241	220	281	306	90			
(24)	123	35	254	136	117	65	130	186	146	84	111	145	213	252	164	87	57	220	188	141	176	96	182		
(25)	134	15	243	112	93	49	119	162	158	95	87	139	202	241	155	98	68	209	177	117	152	119	193	24	

FINAL SOLUTION

NO.	TERMINAL	ROUTES	LOAD	DISTANCE
1	1	T1 4 11 5 T1	51	146
2	1	T1 20 21 8 T1	78	160
1	2	T2 14 3 T2	58	102
2	2	T2 13 19 7 18 T2	95	235
1	3	T3 1 12 15 T3	81	136
1	5	T5 24 2 25 T5	76	125
2	5	T5 17 10 22 T5	76	194
3	5	T5 9 23 16 T5	84	306
4	5	T5 6 T5	38	66

TOTAL DISTANCE = 1470

TRUCK AVAILABILITY AND ALLOCATION TABLE

TRUCK	100	90	80	70	60	50	40	30	20
AVAILABLE	2	4	3	2	2	2	4	3	99
ALLOCATED	1	2	3	0	2	0	1	0	0

PROBLEM 9

NUMBER OF DEMAND POINTS =25 NUMBER OF TERMINALS = 5

DEMAND POINT	DEMAND AT THE POINT	DISTANCE TO THE TERMINAL NO. (1)	DISTANCE TO THE TERMINAL NO. (2)	DISTANCE TO THE TERMINAL NO. (3)	DISTANCE TO THE TERMINAL NO. (4)	DISTANCE TO THE TERMINAL NO. (5)
1	20	164	115	62	83	125
2	16	182	56	232	153	268
3	38	182	101	114	98	186
4	24	133	265	153	163	49
5	36	60	201	201	102	116
6	28	147	137	263	143	233
7	18	98	52	143	47	163
8	24	110	93	208	88	195
9	36	91	170	124	71	50
10	22	137	33	153	87	203
11	28	228	153	76	147	174
12	15	137	109	236	116	223
13	30	62	88	149	30	140
14	22	61	131	144	32	91
15	18	39	111	164	44	121
16	25	178	233	64	140	53
17	27	101	89	113	18	132
18	34	151	165	67	72	68
19	23	154	197	74	104	38
20	16	137	81	95	54	152
21	18	66	202	183	103	97
22	26	138	134	65	55	92
23	34	65	162	142	63	69
24	40	81	129	238	118	189
25	17	87	170	262	142	212

TRUCKS

CAPACITY	NUMBER
100	3
90	2
80	4
70	2
60	1
50	2
40	2
30	1
20	99

DISTANCE MATRIX

(2)	171																								
(3)	62	155																							
(4)	174	310	235																						
(5)	166	240	196	89																					
(6)	217	91	227	257	205																				
(7)	81	107	96	210	149	136																			
(8)	168	73	184	237	167	55	90																		
(9)	95	219	146	91	84	183	118	146																	
(10)	91	88	68	249	188	170	40	124	158																
(11)	64	207	53	223	232	279	144	232	160	120															
(12)	189	60	204	265	195	29	107	28	173	139	249	00													
(13)	111	132	126	185	121	114	35	59	93	75	173	86													
(14)	109	177	127	133	70	141	79	104	42	119	174	132	53												
(15)	125	147	143	167	100	111	59	74	72	99	189	102	23	30											
(16)	126	287	178	98	169	273	180	225	90	217	140	253	166	132	164										
(17)	64	143	84	178	117	161	36	106	88	76	128	134	47	49	63	146									
(18)	65	220	227	117	144	212	112	157	60	153	112	185	89	94	114	69	79								
(19)	90	251	152	87	149	244	145	189	64	181	136	217	131	106	136	42	110	32							
(20)	33	137	44	198	152	188	52	140	110	63	92	160	82	83	99	159	40	92	123						
(21)	147	250	172	70	20	213	149	176	66	189	211	204	125	73	103	150	118	126	130	154					
(22)	36	189	94	141	136	198	82	143	59	121	100	111	84	79	99	101	48	34	66	58	118				
(23)	105	212	149	104	59	176	110	138	26	150	170	166	85	34	64	116	82	86	90	113	41	77			
(24)	200	110	216	213	139	74	126	37	146	166	264	65	91	105	75	236	136	187	210	172	146	172	120		
(25)	224	136	240	215	141	75	150	78	170	190	288	76	115	129	99	261	160	211	235	196	149	196	144	41	

FINAL SOLUTION

NO.	TERMINAL	ROUTES	LOAD	DISTANCE
1	2	T2 2 10 T2	38	177
1	4	T4 14 15 T4	40	106
2	4	T4 13 8 12 6 T4	97	289
3	4	T4 25 24 9 T4	93	400
4	4	T4 21 5 23 T4	88	245
5	4	T4 7 17 T4	45	101
1	5	T5 4 16 T5	49	200
2	5	T5 11 3 1 T5	86	404
3	5	T5 20 22 18 19 T5	99	314

TOTAL DISTANCE = 2236

TRUCK AVAILABILITY AND ALLOCATION TABLE

TRUCK	100	90	80	70	60	50	40	30	20
AVAILABLE	3	2	4	2	1	2	2	1	99
ALLOCATED	3	2	0	0	0	2	2	0	0

PROBLEM 10

NUMBER OF DEMAND POINTS =25 NUMBER OF TERMINALS = 5

DEMAND POINT	DEMAND AT THE POINT	DISTANCE TO THE TERMINAL NO. (1)	DISTANCE TO THE TERMINAL NO. (2)	DISTANCE TO THE TERMINAL NO. (3)	DISTANCE TO THE TERMINAL NO. (4)	DISTANCE TO THE TERMINAL NO. (5)
1	25	783	400	252	662	549
2	18	664	248	187	611	579
3	36	693	764	451	151	315
4	24	740	662	290	321	107
5	15	948	906	533	475	137
6	17	998	615	457	812	557
7	28	271	403	686	815	1050
8	23	471	828	638	252	695
9	16	573	468	159	258	331
10	34	756	748	396	251	152
11	22	513	429	783	964	1147
12	18	641	631	294	213	172
13	30	324	556	423	250	539
14	27	258	661	554	350	656
15	16	287	256	346	523	725
16	24	668	552	180	323	217
17	17	417	210	209	449	586
18	28	756	245	297	703	656
19	36	689	181	530	850	926
20	23	385	866	756	372	748
21	22	404	514	302	170	409
22	18	503	598	355	70	364
23	40	182	503	452	390	629
24	25	911	869	497	368	132
25	19	186	733	698	548	852

TRUCKS

CAPACITY	NUMBER
100	4
90	3
80	4
70	2
60	2
50	1
40	3
30	1
20	99

DISTANCE MATRIX

(2)	154																								
(3)	730	656																							
(4)	461	476	293																						
(5)	686	716	343	244																					
(6)	215	367	785	517	655																				
(7)	805	653	936	943	1187	1020																			
(8)	894	808	338	571	679	1064	704																		
(9)	415	364	296	224	468	586	734	502																	
(10)	610	582	158	154	212	671	999	478	284																
(11)	814	662	1085	1040	1284	1029	245	946	834	1114															
(12)	508	480	186	109	309	601	882	483	167	121	997														
(13)	668	567	351	468	715	870	565	272	301	476	780	367													
(14)	810	706	499	590	824	1001	499	201	423	585	741	484	146												
(15)	505	386	644	618	862	720	325	573	409	679	445	562	354	405											
(16)	432	366	301	113	354	499	833	569	125	216	930	114	394	516	508										
(17)	366	247	544	502	723	581	464	608	273	553	561	436	336	459	137	389									
(18)	173	94	748	569	793	378	650	500	456	674	659	572	659	798	478	458	339								
(19)	492	351	944	820	1063	697	498	589	642	926	359	805	730	821	416	710	397	324							
(20)	1012	911	476	691	817	1184	656	145	622	616	898	583	349	205	610	689	664	1003	1026						
(21)	558	476	288	338	546	749	645	336	172	354	794	234	130	253	353	264	294	572	695	454					
(22)	611	541	193	300	515	778	746	290	211	276	895	192	200	309	454	279	383	633	779	410	95				
(23)	649	530	511	558	766	864	425	344	391	574	640	457	144	176	229	484	283	622	645	381	220	321			
(24)	668	683	239	207	110	689	1154	572	388	110	1247	221	608	717	783	317	686	775	1027	710	509	408	729		
(25)	895	776	697	788	1022	1100	384	403	621	783	612	682	344	195	473	714	529	868	833	303	450	507	246	915	

FINAL SOLUTION

NO.	TERMINAL	ROUTES	LOAD	DISTANCE
1	1	T1 7 11 25 T1	69	1314
1	2	T2 19 T2	36	362
1	3	T3 2 18 6 1 T3	88	1126
2	3	T3 17 15 T3	33	692
1	4	T4 3 T4	36	302
2	4	T4 8 20 14 T4	73	952
3	4	T4 23 13 21 T4	92	834
4	4	T4 22 T4	18	140
1	5	T5 5 24 10 T5	74	509
2	5	T5 12 9 16 4 T5	82	684

TOTAL DISTANCE = 6915

TRUCK AVAILABILITY AND ALLOCATION TABLE

TRUCK	100	90	80	70	60	50	40	30	20
AVAILABLE	4	3	4	2	2	1	3	1	99
ALLOCATED	1	2	2	1	0	0	3	0	1

A GRAPH-THEORETIC APPROACH TO DELIVERY PROBLEMS

by

PRAKASH BHATT

B. Tech. (Mech.), Indian Institute of Technology, Bombay, India, 1970

AN ABSTRACT OF A MASTER'S THESIS

submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

Department of Industrial Engineering

KANSAS STATE UNIVERSITY

Manhattan, Kansas

1972

This paper is concerned with the development of a graph-theoretical approach to obtain an optimal or near-optimal solution to both single- and multi-terminal delivery problems. The delivery problems considered in this paper are those which have deterministic demands, symmetrical distance matrices, and different capacity delivery carriers.

A general computer program has been coded in FORTRAN IV for IBM 360/50 to solve the single- and multi-terminal problems with the proposed algorithm. A wide range of computational experiments has been conducted to test the computational feasibility and the quality of the solution produced by the proposed algorithm.

The results obtained shows that this method is competitive with the existing techniques. Although the results obtained for large problems are somewhat inferior to those obtained by the savings approach, it is felt that the imposition of suitable constraints, in addition to distances, on the minimum 1-tree would improve the quality of solution.