A Heuristic-Based Algorithm for Solving Multiple Supply Vehicle Routing Problems

by

Tom Wilger

A thesis submitted to the Faculty and the Board of Trustees of the Colorado School of Mines in partial fulfillment of the requirements for the degree of Master of Science (Mathematics and Computer Science).

Golden, Colorado

Date 4-1-92

Signed: [illegible]
Thomas J. Wilger

Approved: [illegible]
Dr. Ruth A. Maurer
Thesis Advisor

Golden, Colorado

Date 4\2\92

[illegible]
Dr. Ardel J. Boes
Professor and Head
Department of Mathematics and Computer Science

Abstract

One class of integer programming problems is the vehicle routing problem. Traditionally, optimal solutions to vehicle routing problems have been generated using various integer programming codes and techniques. Unfortunately, due to the high computational complexity of the algorithms, these methods fail to yield solutions for large problems in a reasonable amount of time. In light of this dilemma, many industrial applications use heuristic algorithms to generate near optimal solutions, usually in linear or polynomial time. This thesis presents such an approach. Further, the methods presented here will be utilized to solve a specific type of vehicle routing problem in which delivery of the largest possible quantity of resources takes priority over meeting customer demand. The algorithm produced will be efficient with respect to time complexity, so that for even large problems, perhaps up to 50 demand points, it may be implemented on a PC based system.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Acknowledgments

I would like to thank all those who helped me in the research, and writing of this thesis. I would like to especially thank my advisor, Dr. Ruth Maurer for her time and dedication to this project, and Dr. Woolsey for his support in the entire degree process. I would also like to thank my friend Julia Long who helps keep me focused on my work, and my wife Jennifer, who supported me in this endeavor. I extend my thanks to the guild for their support, and Golden Software for their help with generating tables and figures.

Introduction

In nearly every manufacturing situation, the manufacturer is faced with the problem of how to best distribute its goods to customers while minimizing transportation costs. Although many manufacturers pass on shipping costs to low volume customers, producers must still keep costs down in order to remain competitive in the marketplace. With goods such as gravel or sand, where the cost of distribution far outweighs the actual production cost, optimization of shipping routes can mean the difference between a sand or gravel crusher receiving a bid or losing it to a competitor.

Fortunately, many of these routing problems are relatively small, especially when they involve the transport of raw materials. In such applications, goods must be transported from a single supply point to only a few demand points--and, in some cases, to only one demand point. These simple cases do not require a routing algorithm, the implementation of which would be considered "overkill" in many such situations. However, with even as few as five demand points, routing trucks from a single supply point becomes increasingly difficult.

Consider a situation in which six customers, randomly spaced, require an equal amount of a given product. Suppose the producer of this product has 2 trucks, each with the capacity to meet the demands of 3 customers. There are $\binom{6}{3} = 20$ possible ways to assign the 2 trucks to the 6 customers. For each of these 20 ways to assign the 2 trucks, there are $3! = 6$ ways to route each truck to its 3 customers. Thus in all, there are $\binom{6}{3}(3!) = 120$ possible ways to route the 2 trucks so as to satisfy the customers' demands. By increasing the number of customers in our example to 10, and assuming the same number of supply trucks, the solution space grows to $\binom{10}{5}(5!) = 30240$ possibilities. Route optimization algorithms were designed for problems as large as and larger than these.

This thesis presents a heuristic-based vehicle routing algorithm which may be used to solve the type of problem described above. This algorithm will be compared to other algorithms designed to solve similar problems. Examples, both theoretical and actual, will be presented to help validate the algorithm's solution.

Traveling Salesman

The vehicle routing problem discussed above is closely related to a similar problem, the traveling salesman problem, which has been studied extensively by both computer scientists and mathematicians. The problem statement is as follows:

> "A salesman, starting from a city, intends to visit each of (n-1) other cities once and only once and return to the start. The problem is to determine the order in which he should visit the cities to minimize the total distance traveled, assuming that the direct distances between all city pairs are known."[1]

To date, a deterministic algorithm has not been established that can solve this problem in polynomial time.[2] A polynomial time algorithm is an algorithm in which the amount of computation required to solve a problem of size N is directly proportional to some polynomial function of N.[3] However, this problem has been solved in polynomial time by using a non-deterministic algorithm. The combination of this non-deterministic solution and the lack of a deterministic polynomial time algorithm causes the traveling salesman problem to be classified as NP - Complete.[4]

Even though the amount of computation required for large problems of this type makes complete solution difficult, there exist many algorithms which provide good approximations to the optimal solution. One such method which can be used to approximate the optimal solution to NP-complete problems, particularly the traveling salesman problem, is simulated annealing. The process of annealing has been around in the physical world for some time. Webster's defines annealing as "to heat and then cool (as steel or glass), usually for softening and making less brittle." In the purifying of physical substances, steel or glass may be heated to a high temperature and then slowly cooled so as to separate, or anneal, the unwanted substances from the product.[5] In optimization theory the annealing process is similar. The search for an optimal solution is guided by a control parameter, comparable to temperature in the physical process. This parameter starts out large, and is gradually lowered. The purpose of varying the parameter in this way is to avoid being caught in local minim of the solution space. Howell describes this algorithm as applied to NP-Complete problems as follows.

```
Initialize
    The system configuration.
    The cost of the system configuration.
    The current control parameter, temperature.
Repeat:
    Randomly alter the current system configuration, obtaining a
    (potentially) new configuration.
    Evaluate the new cost of this "candidate" system configuration.
    If new cost < old cost then
        accept the candidate system configuration as the current
        configuration.
    else {new cost ≥ old cost}
        begin
        Select a random number (between 0 and 1).
        If the random number is less than a temperature-dependent
        quantity ($r < e^{\frac{(new\ cost\ -old\ cost\ )}{temperature}}$) then
            accept the new configuration as the current
            configuration.
        else
```

```
                    keep the old configuration as the current
                    configuration.
               end
          Lower temperature.
     Until undetermined number of iterations.[6]
```

As illustrated above, this algorithm has no stopping criterion. A stopping criterion may be specified by the user, perhaps by causing the program to exit when it has found a solution which exceeds, by a certain percentage, some specified lower bound. To provide a better understanding of this algorithm, consider the following problem. A "Miracle Grow" hair tonic salesman wishes to sell his product in the following five cities: A, B, C, D, and E. Since profit equals revenue minus cost,[7] and this hair tonic salesman has a passionate love for money, he desires to find the least cost tour starting at his home town which includes all five cities and then returns to his home town. (see figure 1) Unfortuanately, since this salesman is a fraud (his hair tonic is only a mixture of alcohol and water), he cannot visit any city more than once for fear of being attacked by an angry mob of dissatisfied, bald customers.

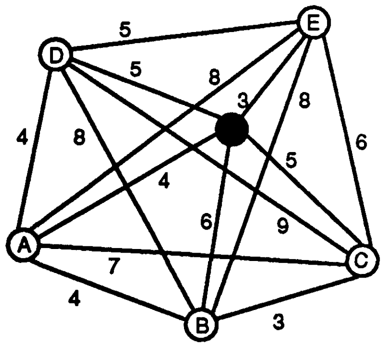


Figure 1

Hair Tonic Salesman Network

For this example, the inital configuration of the problem will be set to: S - A - B - C - D - E - S. This particular tour yields a cost of 28. Further, the inital control parameter *temperature* will be set to 5. Following the algorithm above, a possible new configuration is randomly generated, say S - B - A - C - D - E - S. The *new cost* for this candidate configuration is 34. Since *new cost* ≥ *old cost* a random number between 0 and 1 is generated, say *r*=0.29. The temperature dependent quantity $q = e^{\overline{\frac{34-28}{5}}} = 0.301$ is greater than *r*. The new configuration S - B - A - C - D - E - S with a cost of 34 is thus accepted as the current configuration with a *cost* of 34. Had q been less than or equal to r, the old configuration would have remained as the current configuration. The variable *temperature* is now lowered arbitrarily to say 4 and the process repeats.

When Howell tested this algorithm on a tour of 64 cities randomly spaced on a 100 X 100 grid, the algorithm, whose initial solution yielded a cost of 4115.79, produced a solution with a cost of 558.10 after only 40,000 iterations.

To determine whether the simulated annealing algorithm could find an "obvious" global minimum, the cities to be toured were arranged equally spaced in a circle. The algorithm started with a randomly generated solution, and, after 30,000 iterations, found the global optimal solution .[8]

Routing Algorithm

The algorithm devised to solve the particular class of routing problems presented to the author is based on a vehicle routing algorithm presented by Clarke and Wright in the July - August 1964 issue of Operations Research. As introduced by Clarke and Wright, the algorithm was intended to provide solutions to routing problems involving both multiple demand points and multiple trucks dispatched from a single supply point.

As an illustration of the single supply, multiple demand point algorithm offered by Clark and Wright, consider the following supply - demand network.

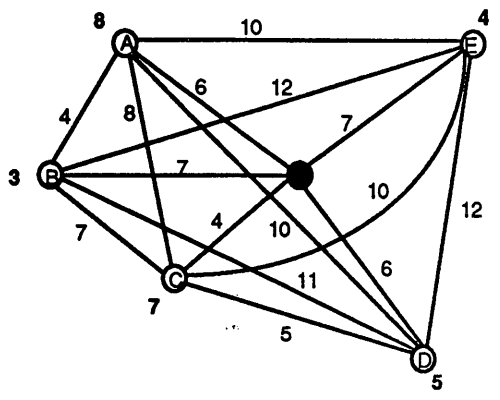


Figure 2

Single Supply Point Network

The lettered circles represent demand points, the solid circle represents the supply point, and the bold numbers depict the number of product units desired by each given demand point. Further, assume that the supply point has 2 trucks available, each with a capacity of 15 product units.

First, all distances between pairs of points, whether two demand points or a demand and a supply point, must be determined. For a problem with n customers and 1 supply point, this information can be stored in an $(n+1) \cdot (n+1)$ matrix with the distance from point i to j being the ijth entry in the matrix, whether this algorithm is being implemented by hand or through a computer. One advantage of this method is that it does not require the distance from point i to j be the same as the distance from j to i. This allows the algorithm to take into account such variations as unequal travel time between two points due to uphill or downhill sections of the route. Also note that when using this method, distances may be replaced with cost, or travel time without any modification to the algorithm.

The next step in the algorithm is to build a table which depicts the potential distance saved in the overall routing schedule if two points p and k were combined in a truck's route, as opposed to requiring a return to the supply point between customers p and k. Consider the network example in Figure 2. Table entry A,B is generated as follows: The distance from the supply point to customer A is 6, thus the total distance traveled by a vehicle from the supply point to customer A and back is 12. Likewise, since the distance from the supply point to customer B is 7, the round trip distance from the supply point to customer B and back is 14. Given that the distance from A to B is 4, the distance saved by combining customers A and B into one route is: $(2(6)+2(7))-(6+7+4)=9$. Repeating this process for all pairs of customers results in the following distance saved table.

Table 1

Single Supply Distance
Saved Table

	A	B	C	D	E
A	-	9	2	2	3
B	9	-	4	2	2
C	2	4	-	5	1
D	2	2	5	-	1
E	3	2	1	1	-

At this point, routings may be established. First, the scheduler must find the largest entry in the distance saved table. In this example, the largest number, 9, corresponds with the entry A,B. Next, the scheduler must determine whether the first truck has a large enough capacity to supply points A and B. Since Truck 1 has an initial capacity of 15, the route A - B is assigned to Truck 1. If Truck 1 did not have enough available capacity to supply both point A and point B, the algorithm would have chosen another pair of points, or sent Truck 1 to either point A or point B alone. Truck 1's available capacity is now reduced by the amount it must deliver to customers A and B. Therefore, the available capacity of Truck 1 is 15 - (8+3) = 4. The algorithm again searches the distance saved table for the largest entry which corresponds to a route adjacent to either endpoint of the truck's current route, and which does not exceed the truck's current resources. Since Truck 1 has an available capacity of only 4 product units, the algorithm will select the table entry corresponding to customer points E and A. With the addition of this new customer to Truck 1's route, the truck's available capacity decreases by 4. Thus the final routing for Truck 1 is S - E - A - B - S, finishing with an available capacity of 0.

The algorithm continues with the above method until all trucks are routed or all customers satisfied.[9] In this example, when the algorithm terminates the following routing schedule is established:

Table 2

Single Supply
Truck Routing Schedule

Truck #	Route	Total Distance	Total Load
1	S - E - A - B - S	28	15
2	S - C - D - S	15	12

This method of routing trucks works quite well when limited to problems which employ only one supply point from which trucks are being dispatched. However, this limitation reduces the algorithm's application to many real world problems. In the fuel industry, for example, tanker trucks often carry fuel to customers all over the country with trucks originating from many different locations where fuel has been stored. Therefore, a method is needed which will route trucks originating from multiple points of supply in different geographic locations.

Before determining whether Clarke and Wright's approach could be modified to address such complications, it was necessary to discover how the original algorithm actually worked. During this process, the truck routing problem was conceptualized as a series of tree search problems, having as the goal of each search the maximum distance saved by combining various customers into a truck's route. Consider the routing problem in Figure 2. This problem could be solved by finding the path which gives a feasible routing schedule with the largest sum of distance saved arcs.

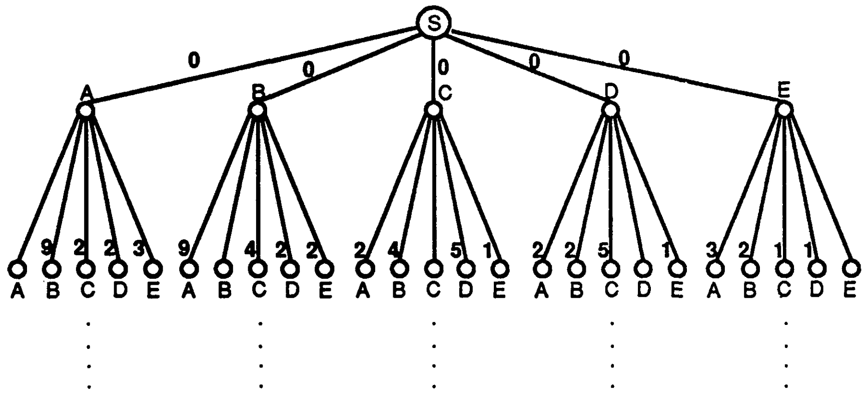


Figure 3

Single Supply Point
Search Tree

As can be seen, this tree is not complete. In order to be complete in this instance, the tree should have another three levels. This completion would require that the bottom level have 3,125 nodes, detail too difficult to show here. However, the use of Clarke and Wright's algorithm here would be analogous to finding the greatest distance saved for the entire route based on the first two levels of the tree. In most cases, the algorithm will cluster a truck's route as tightly as possible by "looking" ahead in the route one customer at a time.

The method used to modify the above algorithm to account for multiple supply points needs to only begin correctly by choosing a supply point relatively close to a customer, and then to let the algorithm add additional customers to that particular route. Unfortunately, this method increases the amount of data storage required, as it requires a distance saved table for each supply point. Thus, if the example in Figure 2 were modified to contain two supply points, two distance saved tables would have to be created, one for each supply point.

Methods Considered

Many methods have been published to solve both theoretical and real world vehicle routing problems. In fact, this type of problem interests not only the operations research community, but the mathematics community as well. These published methods vary in the degree to which they produce optimal solutions, and thus require different amounts of computational resources. For the purposes of this thesis, as basis of comparison, worst case, and sometimes average case time complexity of each algorithm represented in O - (read "big O") notation will be considered. Although in some instances worst case complexity may be a bit misleading, it is easier to calculate than average case time complexity and seems to work fairly well. For the remainder of this work, then, $O(g(n))$ shall be defined as follows:

> **Definition:** A function f(n) is said to be $O(g(n))$ if and only if there exists a natural number N and a real number C, such that for all $n \geq N$, $Cg(n) \geq f(n)$.[10]

In short, the reason for using the above notation when discussing algorithm time complexity is to provide a means of classifying algorithms into groups. This grouping attempts to give an intuitive understanding of how an algorithm performs apart from such biases as problem size or computer clock speed. $O(g(n))$, as defined above, also provides an upper bound on the time complexity function f(n) of a given algorithm. Thus, using such notation it is meaningful to say that an algorithm having a time complexity of $5n = O(n)$, is better than an algorithm producing the same result whose time complexity is $2n^2 = O(n^2)$. Without such a tool, the above statement would be incorrect unless a specific range of n were specified.

One of the more straightforward approaches to solving this class of problems employs the methods of integer programming. Consider the following simple customer/supplier network.

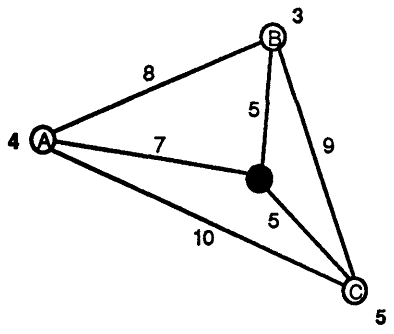


Figure 4

Integer Programming
Single Supply Point Network

As in the first example, the open circles represent customers, with the bold numbers next to them indicating the number of units desired by that customer. The solid circle represents the supply point and the numbers by each arc show the distance (or perhaps cost incurred) to travel from one node to the other. If 2 trucks are used, each with a capacity of 8 units, the problem could be formulated in the following integer program:

Assumption:

1) Paths may only be traveled once per truck.

Variables:

R_{tik} = 1 if truck t travels from point i to point k, 0 otherwise.

S_{ti} = amount supplied by truck t to point i.

Objective Function:

Minimize:
$$\begin{aligned} Z = {} & 7R_{1SA} + 5R_{1SB} + 5R_{1SC} + 8R_{1AB} + 10R_{1AC} + 9R_{1BC} + \\ & 7R_{2SA} + 5R_{2SB} + 5R_{2SC} + 8R_{2AB} + 10R_{2AC} + 9R_{2BC} + \\ & 7R_{1AS} + 5R_{1BS} + 5R_{1CS} + 8R_{1BA} + 10R_{1CA} + 9R_{1CB} + \\ & 7R_{2AS} + 5R_{2BS} + 5R_{2CS} + 8R_{2BA} + 10R_{2CA} + 9R_{2CB} \end{aligned}$$

Constraints:

Continuous Route - Ensure that a continuous route is maintained.

$$R_{1SA} + R_{1BA} + R_{1CA} - R_{1AS} - R_{1AB} - R_{1AC} \leq 0$$
$$R_{1SB} + R_{1AB} + R_{1CB} - R_{1BS} - R_{1BA} - R_{1BC} \leq 0$$
$$R_{1SC} + R_{1AC} + R_{1BC} - R_{1CS} - R_{1CA} - R_{1CB} \leq 0$$

$$R_{2SA} + R_{2BA} + R_{2CA} - R_{2AS} - R_{2AB} - R_{2AC} \leq 0$$
$$R_{2SB} + R_{2AB} + R_{2CB} - R_{2BS} - R_{2BA} - R_{2BC} \leq 0$$
$$R_{2SC} + R_{2AC} + R_{2BC} - R_{2CS} - R_{2CA} - R_{2CB} \leq 0$$

Truck Capacity - Ensure that trucks' capacity is not exceeded.

$$S_{1A} + S_{1B} + S_{1C} \leq 8$$
$$S_{2A} + S_{2B} + S_{2C} \leq 8$$

Customer Demand - Ensure that customer demand is met.

$$S_{1A} + S_{2A} \geq 4$$
$$S_{1B} + S_{2B} \geq 3$$
$$S_{1C} + S_{2C} \geq 5$$

Turn on Route - If S_{ij} is on at any level, then some R_{ixj} must be on.

$$S_{1A} - 8R_{1SA} - 8R_{1BA} - 8R_{1CA} \leq 0$$
$$S_{1B} - 8R_{1SB} - 8R_{1AB} - 8R_{1CB} \leq 0$$
$$S_{1C} - 8R_{1SC} - 8R_{1AC} - 8R_{1BC} \leq 0$$

$$S_{2A} - 8R_{2SA} - 8R_{2BA} - 8R_{2CA} \leq 0$$
$$S_{2B} - 8R_{2SB} - 8R_{2AB} - 8R_{2CB} \leq 0$$
$$S_{2C} - 8R_{2SC} - 8R_{2AC} - 8R_{2BC} \leq 0$$

As seen above, the integer program formulation is quite large, even for a problem of relatively small size. Further, this type of formulation increases exponentially as the number of customers increases, and requires an enormous amount of computation to solve. However, the advantage of using this method is that, once solved, it will produce an optimal routing schedule.

Many different integer programming algorithms could be used to solve the above problem. One of the more common approaches is the Branch and Bound algorithm. This algorithm is similar to an implicit enumeration of all feasible solutions, but in many cases it may eliminate certain branches of the search tree by using the cost of the current solution as an upper bound for the cost of other routes being evaluated. Suppose that a routing which yields a cost or distance of x has been found. Continuation along a path whose cost was greater than x would then prove useless. This part of the search tree could be then pruned, thereby avoiding the enumeration of certain routes.[11] Since there is no guarantee that this algorithm will find feasible integer solutions early enough in the process significantly prune the search tree, the worst case time complexity of the branch and bound algorithm is indeed a worst case, namely $O(2^n)$, where n is the number of variables in the problem formulation. However, since there is usually a significant amount of pruning to be done, the average case time complexity is somewhat better. The basic methodology of the branch and bound algorithm can be represented by a flow chart. (see Figure 5)

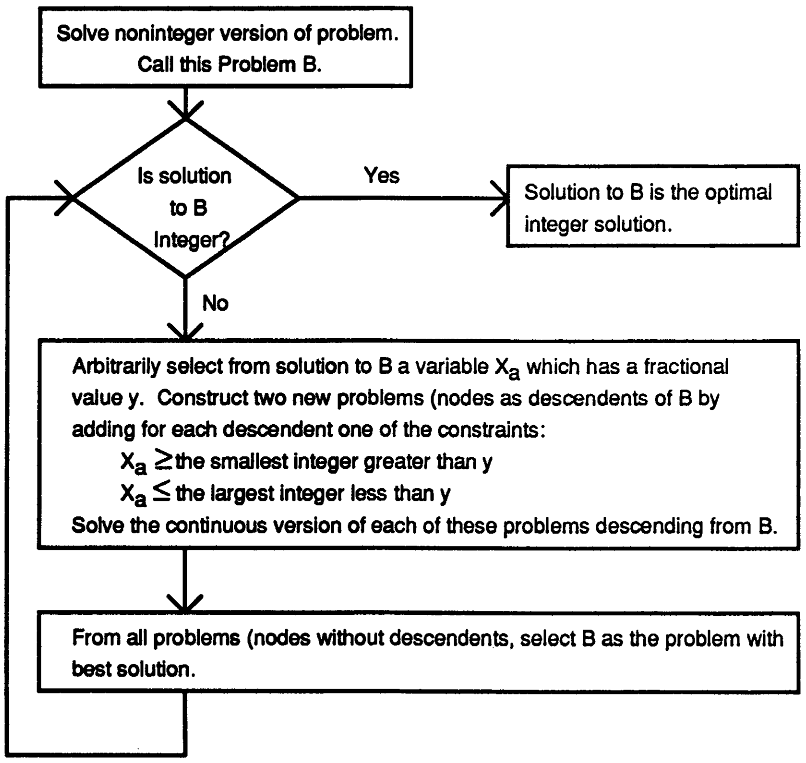

Solve noninteger version of problem. Call this Problem B.
Is solution to B Integer?
Yes
Solution to B is the optimal integer solution.
No
Arbitrarily select from solution to B a variable X_a which has a fractional value y. Construct two new problems (nodes as descendents of B by adding for each descendent one of the constraints:
$X_a \geq$ the smallest integer greater than y
$X_a \leq$ the largest integer less than y
Solve the continuous version of each of these problems descending from B.
From all problems (nodes without descendents, select B as the problem with best solution.

Figure 5

Branch and Bound
Flowchart

To better understand this algorithm, it is best to consider a smaller example than the routing problem stated above.

Objective Function:

Minimize: $Z = X_1 + 3X_2$

Constraints:

$$X_1 \leq 4$$
$$X_2 \leq 5$$
$$2X_1 + 4X_2 \leq 18$$
$$X_1, X_2 = 0,1,2,\ldots$$

Initially, the top node of the search tree represents the original problem in its non-integer form. As shown in Figure 6, the solution to this problem generated by the simplex algorithm is not an integer. Since this is a maximization problem, the solution to the integer problem is bounded above by $Z_1 = 82.7$. This problem will be referred to as problem B. A variable is now selected, say X_1, and the problem is partitioned into two parts; one problem with the added constraint $X_1 \leq 1$, and the other with the added constraint $X_1 \geq 2$. Using the Simplex algorithm, these two descendants of B are now solved and their solutions noted. The value of the objective function of the right child of node 1 is $Z_3 = 80$, the objective function for the left node, node 2, $Z_2 = 81.5$. Thus according to the flowchart, B is set to equal the problem with the best solution, here, node 2. Another variable, X_2, is selected, and problem B is partitioned into two sub problems. The problem represented by node 4 has the additional constraint $X_2 \leq 6$, and node 5 is constrained by $X_2 \geq 7$. The problems represented by nodes 4 and 5 give equally good solutions, $Z = 80$. The solution to node 4, however, is all integer. Thus B is set equal to the problem represented in node 4 and the algorithm terminates. The final solution is thus $Z = 80$, $X_1 = 1$, and $X_2 = 6$.[12] Note that solution 3 gives an equally good Z value; however, in the event of a tie, this algorithm will select the last solution encountered in the search.

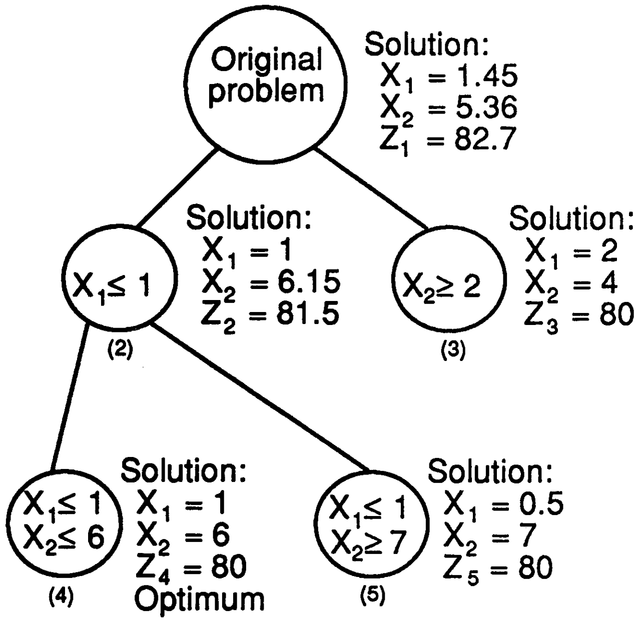

Original problem
Solution: X1 = 1.45 X2 = 5.36 Z1 = 82.7
X1≤ 1
Solution: X1 = 1 X2 = 6.15 Z2 = 81.5
(2)
X2≥ 2
Solution: X1 = 2 X2 = 4 Z3 = 80
(3)
X1≤ 1 X2≤ 6
Solution: X1 = 1 X2 = 6 Z4 = 80 Optimum
(4)
X1≤ 1 X2≥ 7
Solution: X1 = 0.5 X2 = 7 Z5 = 80
(5)

Figure 6

Branch and Bound Tree

Another heuristic algorithm employed to solve this class of problems, along with a generalized assignment problem, was developed and implemented by the University of Pennsylvania and the Information Systems Department of Du Pont in a vehicle routing package called ROVER.[13] ROVER approaches routing problems by dividing them into three sub-problems, and solving each sub-problem by using heuristic and integer programming techniques. The following example (see Figure 7) was presented to illustrate these methods. In the example there are 6 customers and 2 trucks, each truck with a capacity of 30 cubic feet.

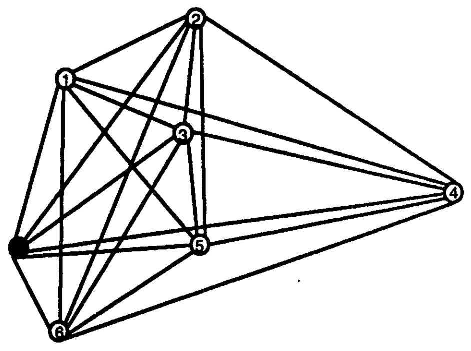


Figure 7

ROVER
Single Supply Point Network

Table 3

ROVER
Inter-Customer Distances

	Terminal	1	2	3	5	6	Demand
1	33						5
2	45	15					6
3	32	14	16				14
4	68	60	51	46			8
5	25	32	36	21	46		9
6	20	48	58	42	65	24	10

As before, Figure 7 represents a collection of demand nodes (open circles) serviced by a single supply node. For purposes of simplicity, all distances are shown in Table 3 rather than on the network. Further, it is assumed that the network is a fully connected graph, thus it is possible to travel from any node to all other nodes directly. The algorithm begins by establishing a set of seed points, or "centers of attraction" for each truck. In ROVER, these seed points can either be set manually or be determined by a set of heuristic rules. In the above example, the authors selected the two seed points using a simple polar sweep procedure.(see Figure 8) Since the total customer demand is 52, and is supplied by two trucks, the average load per truck is 26 cubic feet. Utilizing each customer's demand, the sweep procedure sweeps out 26 units of demand starting with customer 1 and moving toward customer 6. The seed point is then centered between the two rays at a distance equal to the distance from the supply point to the furthest customer in that distance cone.

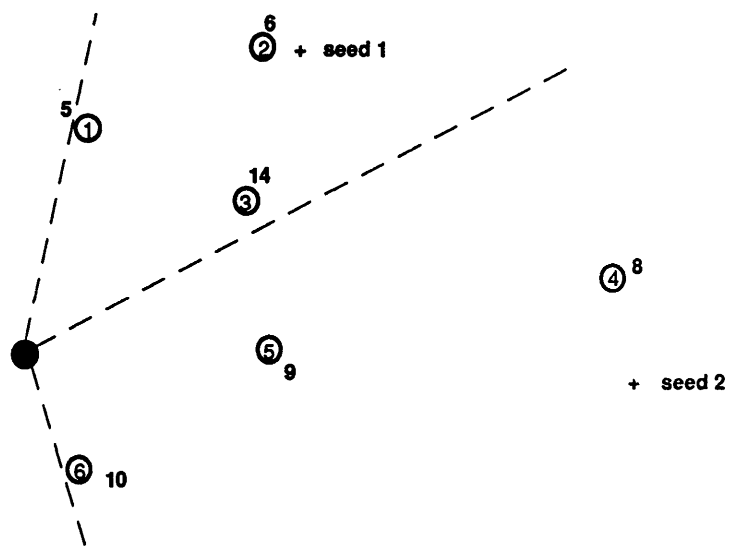


Figure 8

ROVER
Assignment of Seed Points

Next, ROVER calculates the additional distance (or cost) incurred by adding a given customer into the route from the supply point to a seed point and back. (see Table 4)

Table 4

ROVER Insertion Costs

	1	2	3	4	5	6
Seed Loop 1	3	3	6	71	19	32
Seed Loop 2	22	32	13	6	0	13

The algorithm then assigns customers to the two trucks' routes. Here, an integer program formulation which takes the form of a generalized assignment problem minimizes total insertion costs, subject to the vehicles' capacity constraints and customers' demands.

Given:

d_{ik} = cost of inserting customer i into loop k.

a_i = demand for customer i.

b_k = capacity of truck k.

Variables:

y_{ik} = 1 if customer i is assigned to loop k, 0 otherwise.

Objective Function:

$$\text{Min: } Z = \sum_{k=1}^{K} \sum_{i=1}^{n} d_{ik} y_{ik}$$

Constraints:

Each customer assigned to one route

$$\sum_{k=1}^{K} y_{ik} = 1, \; i = 1,\ldots,n,$$

Capacity of truck not exceeded

$$\sum_{i=1}^{n} a_i y_{ik} \le b_k, \; k = 1,\ldots,K,$$

The above example was solved by assigning Customers 1, 2, and 3 to Truck 1 and Customers 4, 5, and 6 to Truck 2. As its final step, this method determines the order in which each truck should visit its customers. ROVER determines this order by employing a heuristic based algorithm developed by Lin and Kernighan.[14] This "Traveling Salesman" algorithm starts with a randomly generated feasible solution (a feasible solution being one in which all customers in the truck's route are visited). It then finds, one at a time, a set of substitute arcs, which, if exchanged with a proper subset of the arcs from the previous solution, would decrease the total distance traveled. This process is repeated until no more exchanges can be made to improve the solution. Lin and Kernighan report that the average time complexity for the algorithm is a little worse than $O(n^2)$. Further, for relatively small problems (less than 42 cities), the probability of achieving an optimal solution on the first trial is very close to 1; this probability decreases slowly to about 0.2 to 0.3 for 100-city problems.[15] If we use this algorithm for our example problem, the routings for Trucks 1 and 2 are found to be S - 1 - 2 - 3 - S, and S - 5 - 4 - 6 - S, respectively.

One noteworthy property of Lin and Kernighan's method is that the solution of the integer program which assigns customers to routes can be found using linear programming techniques. This feature is due to the structure of the problem which has a set of equality constraints all equaling 1. This property is justified similarly to the related transportation problem. Since linear programming techniques can be used to solve the generalized assignment problem and since the other parts of this method utilize heuristic based algorithms, it is assured that a routing solution, even for large problems, can be obtained in a relatively short time. However, since heuristic based algorithms are used both to determine seed points and to solve the traveling salesman problems, the solution generated is not guaranteed to be optimal.

Application

A vehicle routing system which incorporated a heuristic approach for solving multiple supply and demand point routing problems was developed for a local pipe manufacturing company. Thompson Pipe and Steel, Denver, Colorado, manufactures welded steel pipe for various applications. Thompson also builds steel propane tanks--essentially sections of pipe with ends welded to them--for commercial and residential use. The Company is currently manufacturing these tanks at their Kentucky facility, with plans to expand the manufacturing process to their Denver location. They currently have orders for tanks from propane distributors located throughout the United States, and, in fact, are not able to keep up with the demand for this product at their current production rate.

In order to help minimize shipping costs, the Company sought a method of generating a cost effective distribution plan. This method, given customer demand, geographic location, and information pertaining to each manufacturing location's available inventory, would generate a vehicle routing schedule. In order for such a system to be utilized at its maximum potential, it would need to be relatively easy to use, accept input from a variety of sources, and generate a good solution quickly. It was determined that in this instance, ease of use meant that persons with little computer training beyond the selection of items from a menu would be able to operate this program and generate route schedules. Further, to keep data entry costs down, the program should be able to read its input data from existing computer programs, such as a customer order database. Without such a feature, both employees and management may be less enthusiastic about this program's usefulness. In order to ensure that solutions would be generated in a short amount of time, a heuristic based approach to solving the routing problem was developed. One drawback to the heuristic method is that the system would use approximations, and thus could not guarantee optimal solutions.

Since routing problems have concerned industry for some time, computer software has previously been developed to solve the majority of them. Unfortunately, Thompson Pipe's particular situation made it difficult for them to utilize such packages. Due to the nature and current market for their product, a customized system was needed. One such problem arose due to Thompson Pipe's current production limitations--currently, the Company has orders for more propane tanks than they can produce in their existing facility. Thus in creating a routing schedule the Company is more interested in minimizing shipping costs and depleting their tank supplies than minimizing shipping costs while meeting customers' demands. This preference requires a modification of the problem formulation that is not normally encountered. Another complication in Thompson Pipe's situation is that they manufacture more than one size of propane tank. These different tank sizes must be accounted for when determining available truck space. As a partial solution to problem, the requirement was set up that a customer may only be supplied by one truck. Thus the Company may not split a customer's order between two or more trucks, unless that customer can use each truck's entire capacity. If this constraint had not been placed on the solution space, a cost effective routing would have to be determined, but the Company would also be faced with the problem of loading the trucks with different size tanks so as to minimize total wasted space on the fleet of trucks. This problem, which is sometimes referred to as the "knapsack problem," is of the same class as the traveling salesman problem and thus could be very computationally expensive to solve.

Implementation

As was mentioned in Chapter 2, the multiple supply vehicle routing algorithm developed for this specific application is based on a simpler, single supply point version established by Clarke and Wright. To provide a means of comparison between this algorithm and others used to solve similar problems, a flowchart of the algorithm (see Appendix A) and time complexity analysis are provided. As shown in the flowchart, the main body of the program is broken into three parts: the generation of distance saved tables, the routing of supply trucks to customers utilizing one or more complete loads, and establishing routes to supply customers utiliizng only partial loads.

For the purposes of analyzing this algorithm, the following variables are defined:

Variables

n, the number of customers to be considered in the schedule.
k, the total number of supply vehicles from all supply points.
s, the number of supply points to be considered.

In generating the distance saved tables, one table must be created for each supply point, thus **s** tables are created. As shown in the flow chart, each individual operation is accomplished in *O*(1) time. However, since each table has n^2 elements and there are **s** tables, this section of the algorithm is executed in $O(sn^2)$ time. The next step in the algorithm determines which customers are able to utilize one or more entire loads of product. For each of the **n** customers, all **k** supply vehicles must be searched to see if there exists one vehicle whose capacity is less than or equal to a given customer's demand. To perform this operation in a worst case scenario for all customers requires computation proportional to $O(kn)$. The final step in the program routes the remaining vehicles to customers who can only use a partial load. In a worst case situation, this last section has the highest time complexity of all parts of the program. As seen in the flowchart, determining the initial route for a vehicle requires $O(kn^2)$time. However, if the program was presented with a situation whereby only two customers could be served by every vehicle, this process would have to be repeated 1/2(n) times, thus yielding a time complexity for this section of $O(kn^3)$. Because this is the highest order time complexity of any part of the algorithm, by definition of O, it is thus the time complexity of the entire algorithm. Although this time complexity cannot match that of a linear time algorithm, it is better than some of the alternatives presented above which search for the optimal solution. In fact, even with problems of relatively large size (50 or more customers), the program completes execution in a reasonable amount of time. Such a problem, with 50 customers, and two supply points was tested on a 25 MHz 80386 PC. The total run time to generate a solution was just over 35 seconds.

Since the solution of large problems of this type is extremely time consuming, another method is needed to establish that the heuristic algorithm produces at least a reasonable solution. This thesis uses a graphical representation of the customers and supply points, along with arcs which represent the routes generated by the algorithm. Consider the following example (see figure 9).

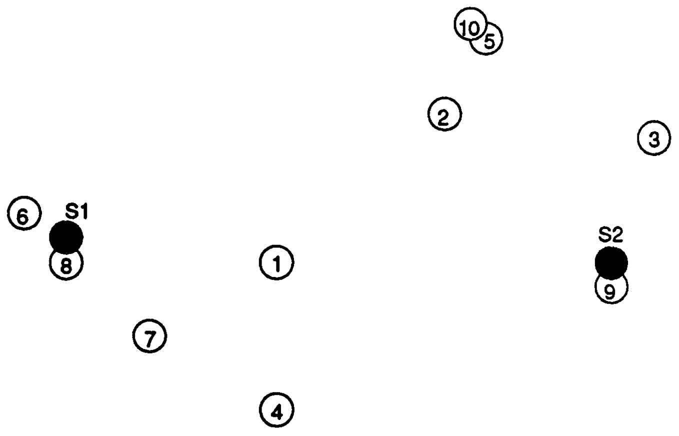


Figure 9

Test Problem 1
Multiple Supply Point Network

As before, the solid circles represent supply points and the hollow circles depict customers. *Again, the arcs between the points are omitted for reasons of clarity and it is assumed that this is a fully connected graph.* In order to represent this network in the program, coordinates must first be assigned to each point. For this example, customer and supply points will be represented in a 20X20 grid. The above problem would then be depicted in the program in the following manner.

Table 5

Test Problem 1
Customer Locations

Customer	X - Coord.	Y - Coord.	Demand
1	6	10	2
2	10	16	7
3	15	15	8
4	6	4	8
5	11	19	2
6	0	12	1
7	3	7	9
8	1	10	5
9	14	9	2
10	11	19	6

Table 6

Test Problem 1
Supply Point Locations

Supply Pt.	X - Coord.	Y - Coord.	Truck #	Capacity
1	1	11	1	15
1	1	11	2	13
2	14	10	3	19
2	14	10	4	15

Once this information is inputed, the program calculates distances between each pair of points and buillds a distance saved table for each supply point. (see Tables 7 and 8)

Table 7

Test Problem 1
Distance Saved Table
From Supply Pt. 1

Cust.	1	2	3	4	5	6	7	8	9	10
1		8.18	9.36	7.70	7.61	0.19	5.33	1.10	10.19	7.61
2	8.18		19.76	6.25	19.94	0.94	3.37	0.48	15.39	19.94
3	9.39	19.76		8.95	21.71	0.68	4.61	0.69	21.63	21.71
4	7.70	6.25	8.95		5.60	0.02	8.83	1.79	12.32	5.60
5	7.61	19.94	21.71	5.60		1.18	2.86	0.35	15.52	25.61
6	0.19	0.94	0.68	0.02	1.18		0.06	0.18	0.25	1.18
7	5.33	3.37	4.61	8.83	2.86	0.06		1.87	6.44	2.86
8	1.10	0.48	0.69	1.79	0.35	0.18	1.87		1.11	0.35
9	10.19	15.39	21.63	12.32	15.52	0.25	6.44	1.11		15.52
10	7.61	19.94	21.71	5.60	25.61	1.18	2.86	0.35	15.52	

Table 8

Test Problem 1
Distance Saved Table
From Supply Pt. 2

Cust.	1	2	3	4	5	6	7	8	9	10
1		8.00	2.80	12.00	7.19	15.82	15.16	16.00	0.94	7.19
2	8.00		7.21	4.56	13.54	10.58	7.21	9.39	0.15	13.54
3	2.80	7.21		0.89	8.93	3.94	2.08	3.23	0.02	8.93
4	12.00	4.56	0.89		3.68	14.14	17.16	15.19	1.57	3.68
5	7.19	13.54	8.93	3.68		10.59	6.47	9.03	0.05	18.97
6	15.82	10.58	3.94	14.14	10.59		19.71	24.91	0.82	10.59
7	15.16	7.21	2.08	17.16	6.47	19.71		20.80	1.22	6.47
8	16.00	9.39	3.23	15.19	9.06	24.91	20.80		0.96	9.03
9	0.94	0.15	0.02	1.57	0.05	0.82	1.22	0.96		0.05
10	7.19	13.54	8.93	3.68	18.97	10.59	6.47	9.03	0.05	

Finally, utilizing the algorithm whose flowchart is given in Appendix A, the program establishs the following routing schedule.

Table 9

Test Problem 1
Vehicle Routing Schedule

Truck #	Route	Total Distance	Total Load
1	S1 - 7 - 8 - 6 - S1	11.73	12
2	S1 - 4 - 1 - S1	19.70	10
3	S2 - 9 - 3 - 5 - 10 - 2 - S2	23.11	19
4	Not Assigned		

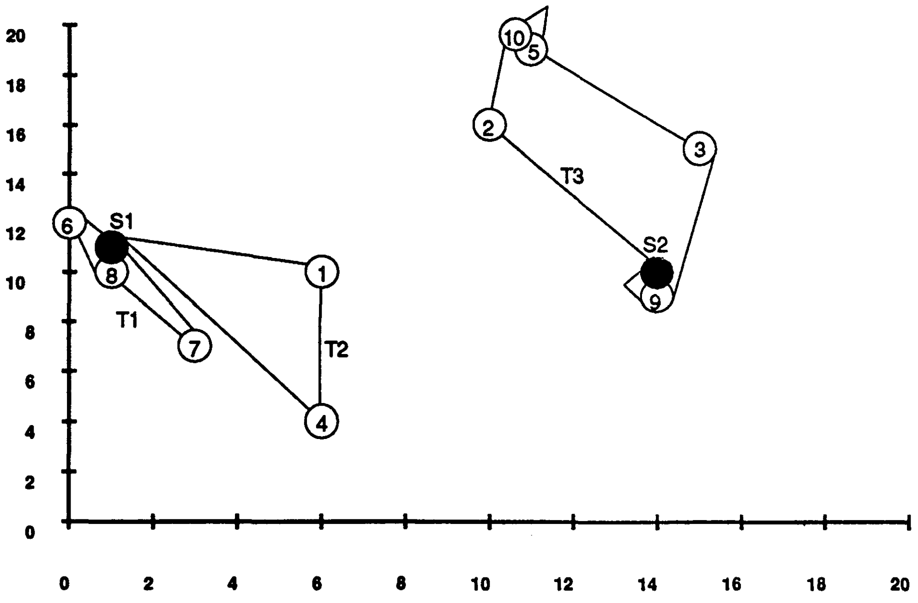


Figure 10

Test Problem 1
Routing Diagram

As a further basis for comparison, two more examples were tested and are presented in Appendix B. The first example consists of 10 customers and 1 supply point with 3 vehicles. The second, an actual problem supplied by Thompson Pipe and Steel, has 8 customers and 1 supply point with 6 delivery vehicles. The second example differs from the first in that customer and supply point locations had to be entered in longitude and latitude coordinates. These coordinates required an additional function be added to the program to calculate distances between two points on the surface of a sphere. For the solutions generated, it should be noted that although the supply of propane tanks has not been exhausted, the demands of all customers may not have been met. This circumstance is due to the restriction that a customer may only be served by one vehicle. Further, the output produced by the program does not specify which customers will have their demands entirely met, and which customers will not. This feature allows the user to decide who will get "preferential treatment" and who will have to wait until the next shipment.

Topics for Further Research

There are many possibilities for further work in modifying the algorithm used for this specific application and in developing vehicle routing strategies in general. In the algorithm created by the author, for example, perhaps a more insightful method for determining which customers should be served by each supply point could be established. This methodology may consider the production of different types of propane tanks at different supply points and the possibility of using other supply points as transshipment points. Another possible way of improving the solutions generated by this algorithm would be to "look ahead" more than two levels in the search tree (see Figure 3). By "looking ahead" instead of searching for the greatest distance that would be saved by combining two customers into a truck's route, the algorithm might be changed so as to find the greatest distance saved by combining three customers into a truck's route. This modification may be hindered by the amount of storage required to hold the three dimensional distance saved tables. Further, by adding this extra dimension, the time complexity of the algorithm would increase due to the extra computation required to search the larger matrices. If taken to its extreme, the idea of "looking ahead" additional levels can show that the algorithm would evaluate all possible solutions, and thus would be equivalent to an exhaustive search algorithm.

Other research which should be considered is whether vehicle routing problems can be proven to belong to the class of NP-complete problems.

Perhaps this research could be accomplished using the concepts of polynomial reducibility. If it could be shown that any instance of a known NP-complete problem may be transformed into an instance of a vehicle routing problem, it would then follow that the class of vehicle routing problems is NP-complete.[16] If it could be determined that vehicle routing problems are of this class, then an algorithm designed to solve any NP-complete problem could be used to solve a vehicle routing problem. To do so would only require finding the correct transformation between the vehicle routing problem and the original NP-complete problem for which the algorithm was devised, applying the transformation to the vehicle routing problem and then generating a solution.

REFERENCES CITED

[1] Phillips, Don T and Alberto Garcia-Diaz. 1990. Fundamentals of Network Analysis. Prospect Heights: Waveland Press, Inc.: 97.

[2] Sedgewick, Robert. 1988. Algorithms. Reading: Addison-Wesley Publishing Company: 635.

[3] Sedgewick, 635.

[4] Sedgewick, 635.

[5] Howell, Russell, W., Simulated Annealing on NP-Complete Problems. Proceedings, ACMS, 1989. 104.

[6] Howell, 104, 105.

[7] Woolsey, R.E.D. Interview with author. Golden Colorado, 27, December 1989.

[8] Howell. 123.

[9] Clarke, G and J. W. Wright. 1964. Scheduling of Vehicles from a Central Depot to a Number of Delivery Points. Operations Research Vol. 12, (July - August): 568 - 580.

[10] Sedgewick, 635.

[11] Sedgewick, 627.

[12] Plane, Donald and Claude McMillian Jr. Discrete Optimization. Englewood Cliffs: Prentice-Hall, Inc.: 74 - 79.

[13] Fisher, Marshall and Arnold Greenfield, R. Jaikumar, and Joseph T. Lester. 1982. A Computerized Vehicle Routing Application. Interfaces. Vol. 12 Number 4: 42-52.

[14] Fisher, Marshall and Arnold Greenfield, R. Jaikumar, and Joseph T. Lester. 51.

[15] Lin, S. and B. Kerighan. 1972. An Effective Heuristic Algorithm for the Traveling Salesman Problem. Operations Research. Vol. 12: 498-516.

[16] Sedgewick. 636.

SELECTED BIBLIOGRAPHY

Bazaraa, Mokhtar, S., and John J. Jarvis. 1977. Linear Programming and Network Flows. New York: Wiley and Sons.

Luenberger, David, G. 1984. Linear and Nonlinear Programming. Reading: Addison-Wesley Publishing Co.

Phillips, Don T. and Alberto Garcia-Diaz. 1981. Fundamentals of Network Analysis. Prospect Heights: Waveland Press.

Woolsey, R. E. D. 1990. Class Notes - Mathematics Course MA522A - Operations Research, Colorado School of Mines, Golden, CO.

Wu, Nesa and Richard Coppins. 1981. Linear Programming and Extensions. New York: McGraw-Hill.

Appendix A

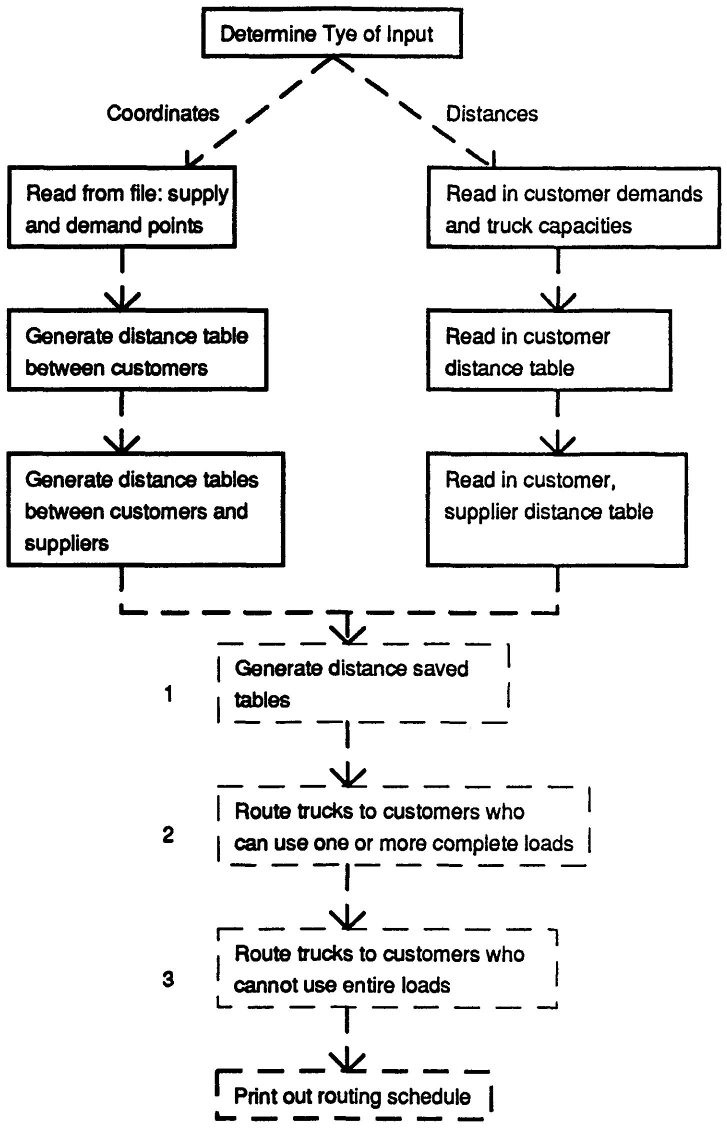
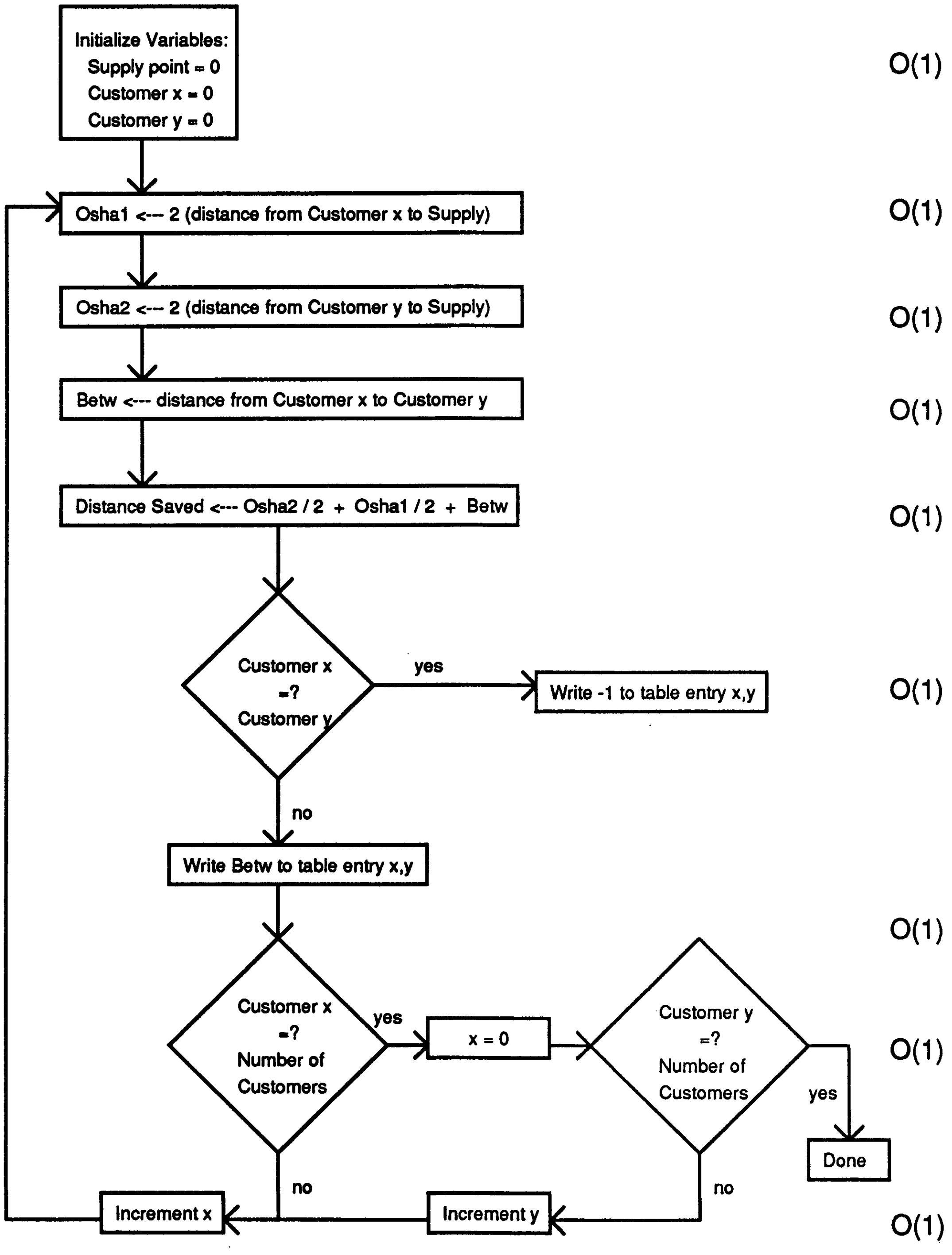

Determine Tye of Input
Coordinates
Distances
Read from file: supply and demand points
Read in customer demands and truck capacities
Generate distance table between customers
Read in customer distance table
Generate distance tables between customers and suppliers
Read in customer, supplier distance table
1
Generate distance saved tables
2
Route trucks to customers who can use one or more complete loads
3
Route trucks to customers who cannot use entire loads
Print out routing schedule

Figure A-1

Vehicle Routing Program
Main Flowchart

Vehicle Routing Program
Distance Saved Tables Flowchart



Initialize Variables:
Supply point = 0
Customer x = 0
Customer y = 0
O(1)
Osha1 <--- 2 (distance from Customer x to Supply)
O(1)
Osha2 <--- 2 (distance from Customer y to Supply)
O(1)
Betw <--- distance from Customer x to Customer y
O(1)
Distance Saved <--- Osha2 / 2 + Osha1 / 2 + Betw
O(1)
Customer x =? Customer y
yes
Write -1 to table entry x,y
O(1)
no
Write Betw to table entry x,y
O(1)
Customer x =? Number of Customers
yes
x = 0
Customer y =? Number of Customers
O(1)
yes
Done
no
no
Increment x
Increment y
O(1)

Figure A-2

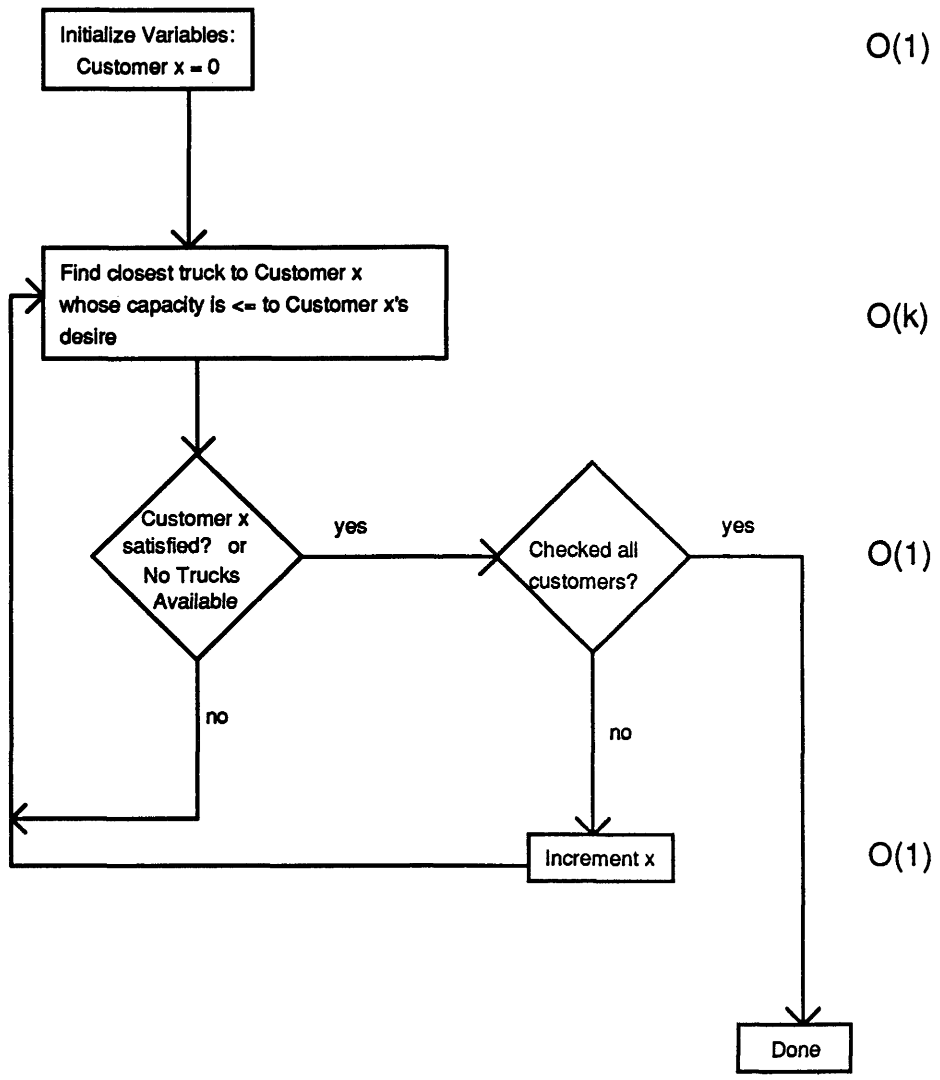

Initialize Variables: Customer x = 0
Find closest truck to Customer x whose capacity is <= to Customer x's desire
Customer x satisfied? or No Trucks Available
yes
no
Checked all customers?
yes
no
Increment x
Done
O(1)
O(k)
O(1)
O(1)

Figure A-3

Vehicle Routing Program
Full Truck Routing Flowchart

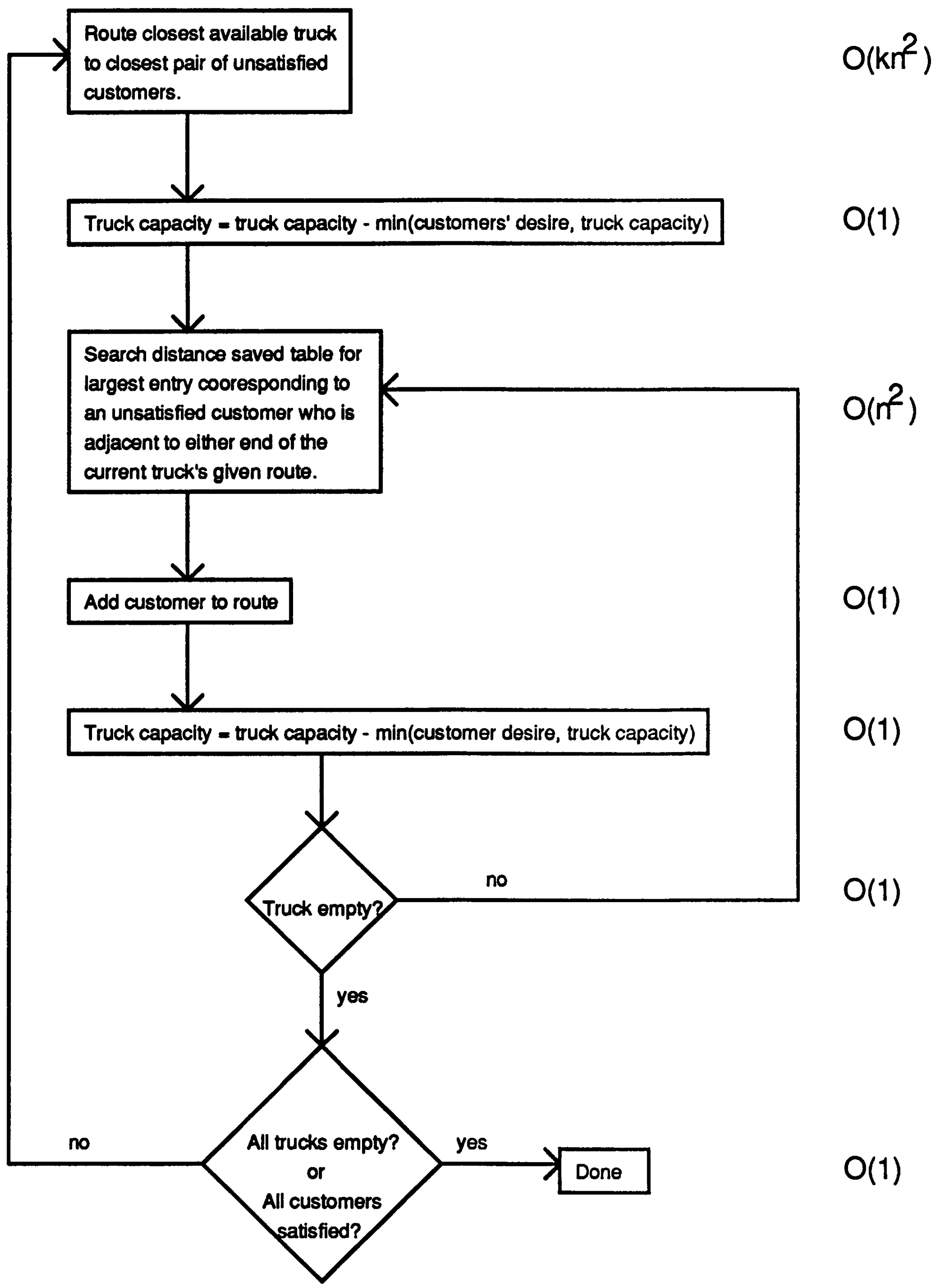

Route closest available truck to closest pair of unsatisfied customers.
O(kn²)
Truck capacity = truck capacity - min(customers' desire, truck capacity)
O(1)
Search distance saved table for largest entry cooresponding to an unsatisfied customer who is adjacent to either end of the current truck's given route.
O(n²)
Add customer to route
O(1)
Truck capacity = truck capacity - min(customer desire, truck capacity)
O(1)
Truck empty?
no
yes
O(1)
All trucks empty? or All customers satisfied?
no
yes
Done
O(1)

Figure A-4

Vehicle Routing Program
Route Trucks Flowchart

Appendix B

Table B-1

Test Problem 2
Customer Locations

Customer	X - Coord.	Y - Coord.	Demand
1	6	10	2
2	10	16	7
3	15	15	8
4	6	4	8
5	11	19	2
6	0	12	1
7	3	7	9
8	1	10	5
9	14	9	2
10	11	19	6

Table B-2

Test Problem 2
Supply Point Locations

Supply Pt.	X - Coord.	Y - Coord.	Truck #	Capacity
1	1	11	1	15
1	1	11	2	13
1	1	11	3	14

Table B-3

Test Problem 2
Distance Saved Table
From Supply Pt. 1

Cust.	1	2	3	4	5	6	7	8	9	10
1		8.18	9.36	7.70	7.61	0.19	5.33	1.10	10.19	7.61
2	8.18		19.76	6.25	19.94	0.94	3.37	0.48	15.39	19.94
3	9.39	19.76		8.95	21.71	0.68	4.61	0.69	21.63	21.71
4	7.70	6.25	8.95		5.60	0.02	8.83	1.79	12.32	5.60
5	7.61	19.94	21.71	5.60		1.18	2.86	0.35	15.52	25.61
6	0.19	0.94	0.68	0.02	1.18		0.06	0.18	0.25	1.18
7	5.33	3.37	4.61	8.83	2.86	0.06		1.87	6.44	2.86
8	1.10	0.48	0.69	1.79	0.35	0.18	1.87		1.11	0.35
9	10.19	15.39	21.63	12.32	15.52	0.25	6.44	1.11		15.52
10	7.61	19.94	21.1	5.60	25.61	1.18	2.86	0.35	15.52	

Table B-4

Test Probelm 2
Vehicle Routing Schedule

Truck #	Route	Total Distance	Total Load
1	S1 - 7 - 8 - 6 - S1	11.73	15
2	S1 - 3 - 9 - 4 - 1 - S1	41.17	13
3	S1 - 3 - 10 - 5 - S1	33.03	14

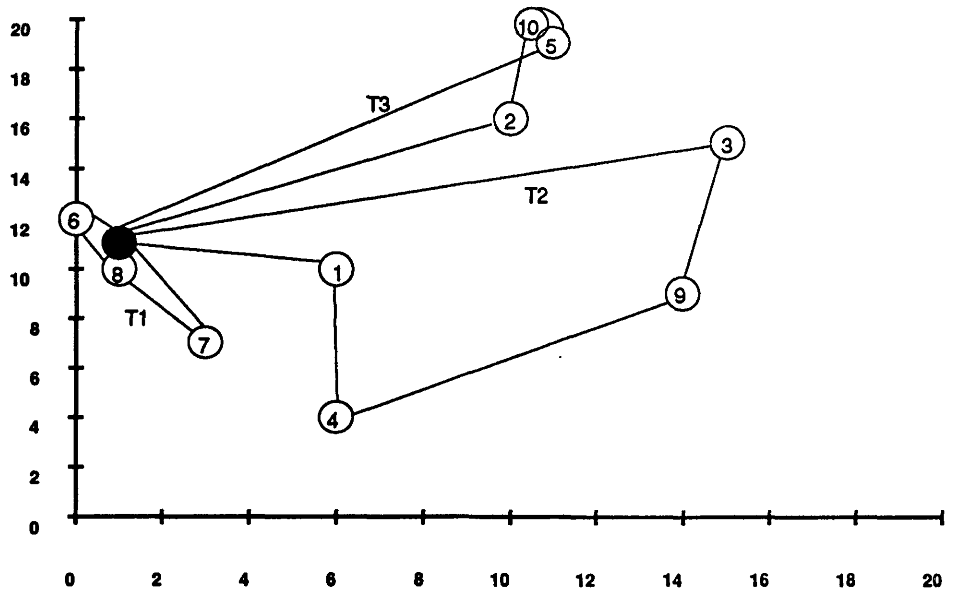


Figure B-1

Test Problem 2
Vehicle Routing Diagram

Table B-5

Thompson Pipe Example
Customer Locations

Customer #	Location	Longitude	Latitude	Demand
1	Fulton, KY	88.8	36.5	14
2	Princeton, KY	87.8	37.2	5
3	Laurel, IN	85.25	39.5	15
4	Princeton, KY	87.8	37.2	11
5	Salem, KY	88.3	37.4	1
6	Big Rapids, MI	85.5	43.7	14
7	Tifton, GA	83.5	31.4	14
8	Salem, KY	88.3	37.4	1

Table B-6

Thompson Pipe Example
Supply Point Locations

Supply Pt. #	Location	Longitude.	Latitude	Truck #	Capacity
1	Princeton, KY	87.8	37.2	1	15
1	Princeton, KY	87.8	37.2	2	15
1	Princeton, KY	87.8	37.2	3	15
1	Princeton, KY	87.8	37.2	4	15
1	Princeton, KY	87.8	37.2	5	15
1	Princeton, KY	87.8	37.2	6	15

Table B-7

Thompson Pipe Example
Distance Saved Table

Cust.	1	2	3	4	5	6	7	8
1		0.00	0.57	0.00	31.40	9.96	67.89	31.40
2	0.00		0.00	0.00	0.00	0.00	0.00	0.00
3	0.57	0.00		0.00	18.74	334.45	97.37	18.47
4	0.00	0.00	0.00		0.00	0.00	0.00	0.00
5	31.40	0.00	18.74	0.00		31.64	3.46	53.40
6	9.96	0.00	334.45	0.00	31.64		67.76	53.40
7	67.89	0.00	97.37	0.00	3.46	67.76		3.46
8	31.40	0.00	18.74	0.00	53.40	31.64	3.46	

Table B-8

Thompson Pipe Example
Vehcile Routing Schedule

Truck #	Route	Total Distance	Total Load
1	S - 3 - S	365.7	15
2	S - 4 - 2 - S	0.00	15
3	S - 6 - 8 - 5 - S	829.44	15
4	S - 7 - 1 - S	875.6	15
5	Not Assigned		
6	Not Assigned		

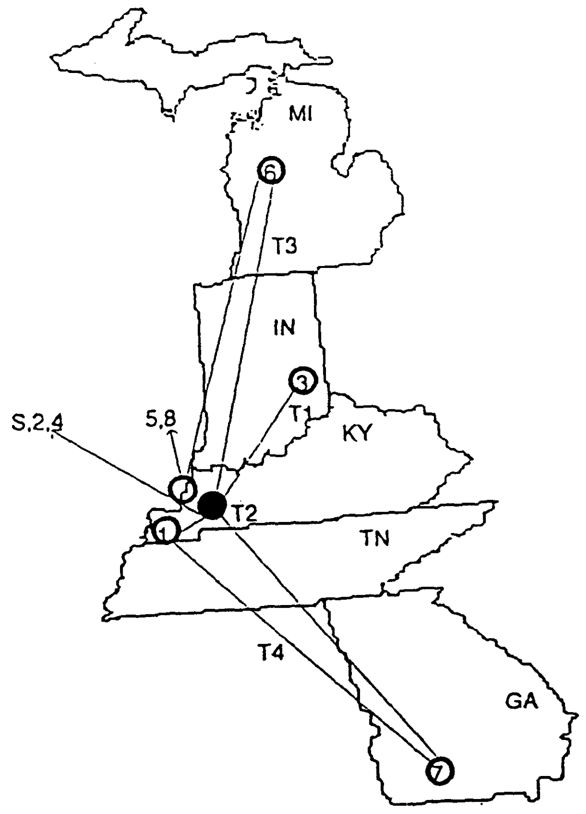


Figure B-2

**Thompson Pipe Example
Vehicle Routing Diagram**