

A PROGRAM FOR SOLUTION OF LARGE SCALE  
VEHICLE ROUTING PROBLEMS

By

JEFFREY ALLAN ROBBINS

Bachelor of Science

Virginia Polytechnic Institute  
and State University  
Blacksburg, Virginia

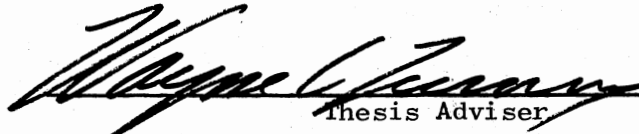
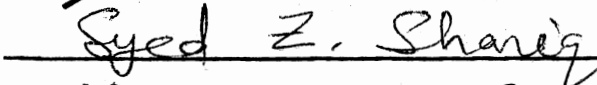

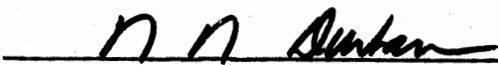
1974

Submitted to the Faculty of the Graduate College  
of the Oklahoma State University  
in partial fulfillment of the requirements  
for the Degree of  
MASTER OF SCIENCE  
May, 1976

AUG 26 1976

A PROGRAM FOR SOLUTION OF LARGE SCALE  
VEHICLE ROUTING PROBLEMS

Thesis Approved:

  
Thesis Adviser  
  
  
  
Dean of the Graduate College

947629

## PREFACE

This research is concerned with the development and implementation of a man-machine procedure to solve large scale routing problems.

A user of the procedure needs no computer programming experience but should be familiar with the routing system under study.

The author wishes to express his appreciation to his major advisor, Dr. Wayne C. Turner, for his guidance and assistance throughout this research and studies leading to the area of routing. Appreciation is also expressed to the other committee members, Dr. Kenneth E. Case and Dr. Syed Z. Shariq for their assistance during the research and in the preparation of the final manuscript.

A note of thanks is given to Dr. David Byrd for his assistance in the computer programming. Thanks are also extended to Dr. Carl B. Estes for allowing the author an opportunity to develop his writing skills, to Margaret Estes for the excellence of the final draft, and to the Industrial Engineering and Management Department for the computer funds to perform this research. In addition, appreciation is given to my parents, Sam and Carol Robbins, for their encouragement and continued interest.

Finally, special gratitude is expressed to my wife, Alice, for the understanding, patience, and encouragement given to her part-time husband.

## TABLE OF CONTENTS

Chapter	Page
I. INTRODUCTION . . . . .	1
Problem Definition . . . . .	1
Objectives of the Research . . . . .	2
Assumptions . . . . .	3
Outline of Thesis . . . . .	4
II. LITERATURE SEARCH . . . . .	5
Introduction . . . . .	5
Traveling Salesman Problem . . . . .	5
Routing Problem . . . . .	8
Summary . . . . .	14
III. THE CLARKE AND WRIGHT ALGORITHM . . . . .	15
Introduction . . . . .	15
The Clarke and Wright Algorithm . . . . .	15
Programming Strategies . . . . .	23
IV. THE TWO-OPT ALGORITHM . . . . .	26
Introduction . . . . .	26
Two-Opt Requirements . . . . .	28
Programming Strategies . . . . .	31
V. COMPUTATIONAL RESULTS . . . . .	35
Introduction . . . . .	35
Background of Problems . . . . .	35
Examination of Results . . . . .	38
Statistical Analysis . . . . .	40
Additional Computational Experience . . . . .	42
VI. MAN-MACHINE INTERACTION . . . . .	46
Description . . . . .	46
Reasons For Use . . . . .	46
Usage . . . . .	47
Real World Implementation . . . . .	48

Chapter	Page
VII. CONCLUSIONS AND FUTURE RESEARCH . . . . .	53
Conclusions . . . . .	53
Future Research and Extensions . . . . .	54
BIBLIOGRAPHY . . . . .	56
APPENDIX A--USER INSTRUCTIONS . . . . .	58
APPENDIX B--COMPUTER PROGRAM . . . . .	64
APPENDIX C--COORDINATES AND DEMANDS FOR SIX RANDOMLY GENERATED PROBLEMS . . . . .	79
APPENDIX D--USER INSTRUCTIONS FOR INTERACTION PHASE . . . . .	86

## LIST OF TABLES

Table	Page
I. Distance Matrix . . . . .	18
II. Savings Matrix . . . . .	18
III. Ordering of Savings . . . . .	19
IV. List of Problems . . . . .	36
V. Comparison of Algorithms . . . . .	37
VI. Solution Results and Differences for Sign Test . . . . .	41
VII. Execution Times and Differences for Sign Test . . . . .	43
VIII. Solutions to Randomly Generated Problems . . . . .	43
IX. Node Demands . . . . .	60
X. Node Distances . . . . .	60
XI. Data Cards . . . . .	61
XII. Problem 1 . . . . .	79
XIII. Problem 2 . . . . .	80
XIV. Problem 3 . . . . .	81
XV. Problem 4 . . . . .	82
XVI. Problem 5 . . . . .	83
XVII. Problem 6 . . . . .	84
XVIII. Interaction Data Cards . . . . .	88

## LIST OF FIGURES

Figure	Page
1. Branch and Bound Tree . . . . .	7
2. Eight Ways of Connecting Three Arcs . . . . .	12
3. One Vehicle Servicing Each Node . . . . .	16
4. One Vehicle Servicing Two Nodes . . . . .	16
5. Node Locations . . . . .	21
6. Clarke and Wright Route . . . . .	21
7. Flowchart of Clarke and Wright Algorithm . . . . .	24
8. Two Independent Two-Opt Routes . . . . .	27
9. Swapping Arcs Between Routes . . . . .	27
10. System of Two Routes . . . . .	29
11. Replacement of Two Arcs on One of Two Routes . . . . .	29
12. Initial Route and Two Swaps . . . . .	30
13. Sequence of Two-Optimal Changes Within One Tour . . . . .	32
14. Flowchart of Lin Two-Opt Algorithm . . . . .	33
15. Computation Time Versus Problem Size . . . . .	44
16. Storage Requirements Versus Problem Size . . . . .	45
17. Yale School District . . . . .	49
18. Initial Clarke-Wright/Lin Solution . . . . .	50
19. Interaction Solution . . . . .	52

## CHAPTER I

### INTRODUCTION

#### Problem Definition

A well known problem to certain segments of society is the "Transportation Routing Problem" or simply "Routing Problem." This problem can be stated as follows:

Goods are to be distributed from a source to a known set of destinations. These goods are carried by a fleet of carriers of known capacity. An analyst must assign each carrier one or more destinations so that the carriers, starting at the source, deliver the goods to each of the assigned destinations and returns to the source. Each destination is traveled to only once. The objective is to minimize the total distance traveled during delivery. [24, p. 288]

Manifestations of this problem appear in many diverse sectors of the economy. In the public sector, analysts are constantly routing school buses, street sweepers, snow plows, refuse collection vehicles, and other service vehicles. In the private sector, industries route vehicles to serve warehouses or branch stores. In the quasi private sector, the U.S. Post Office Department is faced with a multitude of different routing problems. Finally, many production scheduling problems can be given a vehicle routing formulation. The problem, therefore, is one that is presently receiving a great deal of interest.

The majority of real life situations deal with symmetrical distances. That is, if  $d_{ij}$  denotes the distance between destinations



(nodes)  $i$  and  $j$ , then

$$d_{ij} = d_{ji} \quad \forall i \text{ and } j . \quad (1.1)$$

The asymmetric case allows for the possibility of

$$d_{ij} \neq d_{ji} . \quad (1.2)$$

This is not frequently encountered in real world situations except when the problem involves one way streets or bridges. This research is applicable only to the symmetric case although it would be simple to alter for asymmetric problems.

Regardless of symmetry, all routing problems are combinatoric in nature. This means an exceedingly large number of combinations is possible. Assuming each pair of nodes is linked and distances are symmetric, the total number of different possible routes through  $N$  points is  $\frac{1}{2}N!$ . For example, a group of 12 nodes can be serviced by any one of 239,500,800 routes. As can be seen, exhaustive enumeration is infeasible for all but the smallest problems.

#### Objectives of the Research

Many people have developed procedures, manual and computer, that will optimally or heuristically solve routing problems. Optimal seeking procedures are interesting intellectually but, so far, unrewarding realistically. A problem involving 20 to 30 nodes can be solved optimally but with large storage and computational time requirements. Problems of more than about 50 nodes simply cannot be solved optimally with today's technology.

Heuristic procedures, however, seem to offer hope for good solutions to large scale problems. Thus, the basic objectives of this

research are to explore existing heuristic procedures and attempt to discern those that offer the most promise. Then, the next objective is to efficiently program the chosen technique or combination of techniques in order to produce a procedure that:

- (a) handles "large" symmetric problems,
- (b) produces good solutions,
- (c) is easy and economical to use.

Another objective is to allow the analyst to interact with the solution procedure so that he does not have to blindly accept an answer. This increases the validity of the solution and the faith of the analyst in the final result.

A computer program is developed in this thesis that combines two of the better heuristic procedures into one effective and efficient program. Interaction is allowed and encouraged. Computation results are presented.

#### Assumptions

The following assumptions are made:

- 1) Capacity of each carrier is known. (Usually all carriers have the same capacity but this is not a requirement.)
- 2) Distance between any two nodes, including the source (depot), is known. The distances are usually put into the form of a matrix and must be symmetric.
- 3) Demand at each node is known.
- 4) All carriers or vehicles start and end at the depot. There can be only one depot.
- 5) Each node is serviced once and only once by some vehicle.

The above assumptions are consistent with the problem definition previously given and the literature search to be covered in Chapter II.

#### Outline of Thesis

The results are presented in seven chapters. Chapter I, this chapter, defines the problem and states the objectives of the research. Chapter II reviews the existing literature on Routing Problems. Chapter III discusses the Clarke and Wright route building algorithm and programming procedures. Chapter IV presents the Lin route improvement procedure. Chapter V details the computational experience. Chapter VI discusses the "interaction phase" and its use. Chapter VII presents the summary, conclusions, and ideas on extensions and future research.

## CHAPTER II

### LITERATURE SEARCH

#### Introduction

The vehicle routing problem was probably first formulated by Dantzig and Ramser [9]. In this early paper, they showed the vehicle routing problem to be a generalization of the classical traveling salesman problem (TSP). Since much of the work on vehicle routing problems draws heavily from traveling salesman literature, a brief review is given below and is followed by the review of vehicle routing literature.

#### Traveling Salesman Problem

The TSP is one where a salesman, starting in his home city, wishes to establish an itinerary such that he visits each of  $N$  other cities once and only once and travels a minimum distance. Many solution procedures have been developed but all can be classified as either an "optimal seeking" or "heuristic" procedure.

#### Optimal Seeking Procedures

Optimal seeking procedures are those methods that guarantee an optimal solution. Since the problem is combinatoric in nature, however, optimal seeking procedures require excessive computational

time and storage requirements except for small textbook type problems.

Eastman [10] was the first to solve the TSP exactly. His method is based on a branch and bound strategy where the assignment algorithm is used for computing bounds.

Little, et al. [19], use an approach similar to Eastman's in that bounds are used on the assignment problem. The procedure begins by reducing the associated cost or distance matrix until a zero exists in every row and column. The total reduction is a lower bound on the solution. Penalties are then created for each zero in the matrix by calculating a cost, or penalty, associated with not choosing the corresponding  $(i,j)$  of a zero entry. The zero entry with the highest penalty is then placed in a "tree". The tree (Figure 1) contains two sets of branches. One branch contains all tours including  $(i,j)$ ; the other contains all tours excluding  $(i,j)$ , i.e.,  $(\overline{i},j)$ . As branches are explored, the penalty associated with not traveling  $(i,j)$  is added to the previous lower bound. In addition, as  $(i,j)$  is added to the branching tree, the corresponding row and column are deleted from the cost matrix. A continuation of this "branch and bound" technique results in the optimal tour.

Bellman [2] discusses the formulation of the TSP as a dynamic programming problem. The procedure begins by considering the problem as a "multistage" decision problem. Then, starting at any node and using dynamic programming, Bellman shows the resulting tour to be optimal.

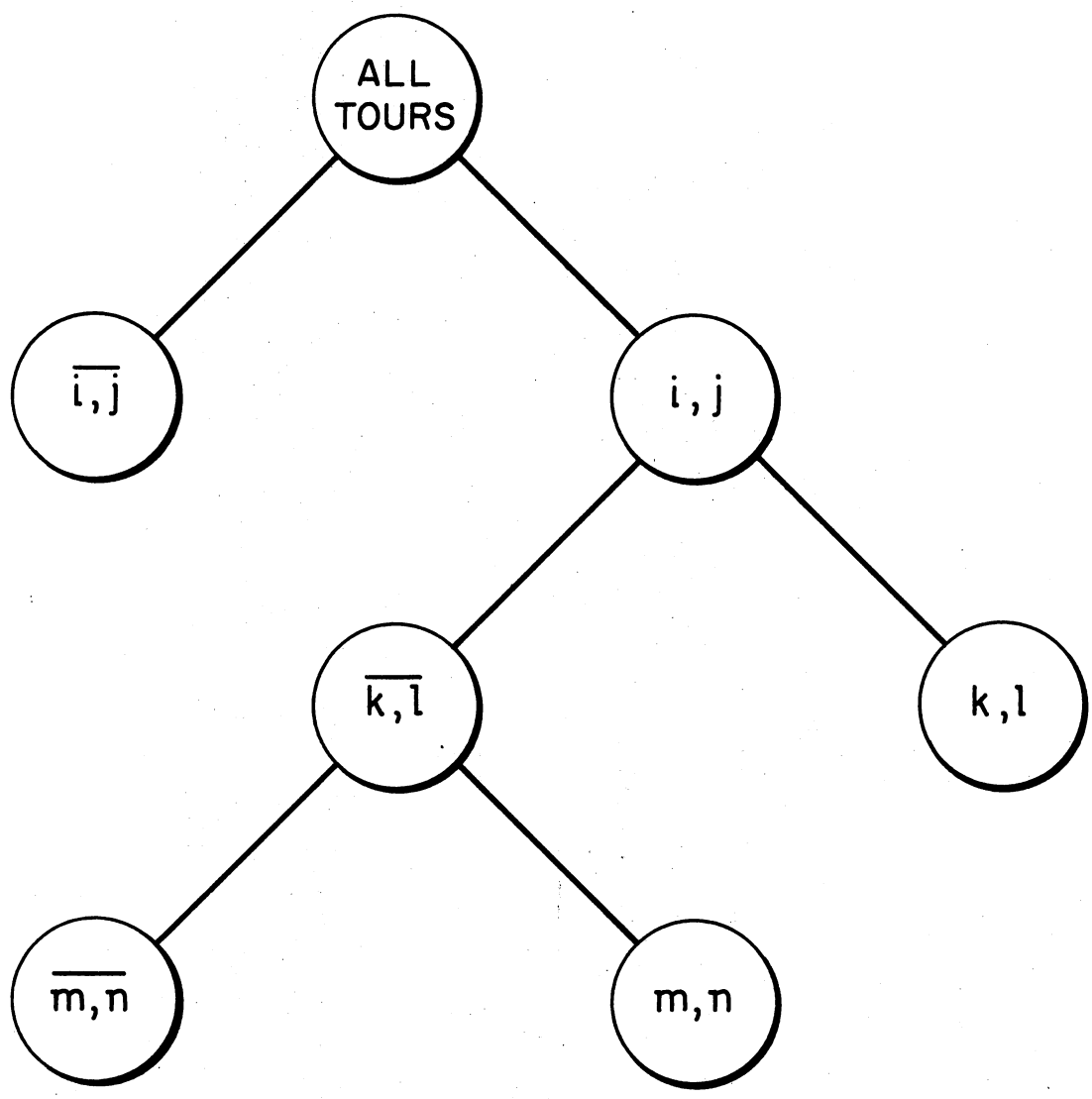


Figure 1. Branch and Bound Tree

### Heuristic Traveling Salesman Algorithms

Bellman, Eastman, and Little, et al., all were able to solve the TSP optimally. However, the computation time and storage requirements increased exponentially in each case. For this reason, only problems with a small number of nodes could be solved. Large problems, approximately 50 nodes or more, must be handled by heuristic techniques. Ashour and Parker [1] suggest using a heuristic where, after starting with some initial node, the nearest unvisited node is traveled to next. This procedure continues until a cycle exists producing the route of the traveling salesman. The procedure is repeated with different initial nodes until all have been used as a starting point. The best of the tours is chosen as the solution.

The most important heuristic, with respect to this thesis, was proposed by Croes [8] and extended by Lin [18]. Croes' algorithm revolves around the idea of removing two arcs from a route and replacing with two different arcs such that the distance of the route is reduced. The Lin algorithm is discussed thoroughly in Chapter IV. Other procedures for solving the TSP, exact and heuristic, have been found and are discussed by Bellmore and Nemhauser [4]. It is recommended as a reference.

### Routing Problem

#### Optimal Seeking Procedures

As previously stated, computation time for solving the TSP increases quite rapidly as the number of nodes increases. This same problem occurs in routing problem algorithms. Still, optimal seeking

routines have been developed. Eilon, et al. [11], use a procedure similar to Little, et al. [19], to optimally solve small routing problems. Svestka and Huckfeldt [22] extended the work of Bellmore and Malone [3] to solve a multiple traveling salesman problem (MTSP), which is very similar to the vehicle routing problem.

There are other optimal seeking procedures, but this research concentrates on heuristic procedures for solving large scale vehicle routing problems. Therefore, the majority of the literature search is in the next section where heuristic algorithms are covered.

### Heuristic Procedures

The solution of a problem with a large number of nodes requires a technique that:

- (a) is fast,
- (b) produces good results, and
- (c) uses reasonable computer time and storage space.

There are many heuristic solution procedures to the Routing Problem that meet at least one and possibly all of the above requirements. The algorithms relevant to this thesis are discussed here. For a more exhaustive survey of the Routing Problem see Turner, et al. [24], or Bodin [5].

As previously mentioned, Dantzig and Ramser [9] were the first to formulate the Routing Problem. In addition to their formulation, Dantzig and Ramser also proposed a heuristic solution procedure. Their heuristic was based on building routes that filled trucks to capacity rather than minimizing the total distance. Clarke and Wright [7] extended this work to consider the minimization of distance as the sole



objective. Clarke and Wright first assume the existence of one carrier for each node. Then, if one truck services any two nodes, the following savings can be calculated for that pair of nodes:

$$S_{ij} = d_{oi} + d_{jo} - d_{ij}, \quad \forall i \text{ and } j, i \neq j \quad (2.1)$$

In the savings equation,  $d_{ij}$  is the distance or cost from node  $i$  to node  $j$  and  $o$  denotes the depot. All savings are then arranged in descending order. The procedure, starting with the pair of nodes having the largest saving, builds routes by combining feasible pairs of nodes in the above order. At every combination, capacity and distance constraints are checked and the procedure continues until all nodes are on a route. A more detailed examination of the Clarke and Wright heuristic follows in Chapter III.

Gaskell [12] proposed a heuristic that is a slight modification of the Clarke and Wright algorithm. He proceeds in the same manner as Clarke and Wright except the savings are calculated as either:

$$\lambda_{ij} = S_{ij} (\bar{D} + |d_{oi} - d_{oj}| - d_{ij}), \text{ or} \quad (2.2)$$

$$\pi_{ij} = S_{ij} - d_{ij} \quad (2.3)$$

where

$$\bar{D} = \frac{\sum_{i=1}^N d_{oi}}{N} \quad \text{and } S_{ij} \text{ is the Clarke and Wright savings.}$$

Gaskell calculations give more weight to nodes with high  $d_{io}$ 's.

Robbins, et al. [21], have shown, using randomly generated problems, the Clarke and Wright method to be at least as good as Gaskell's first savings calculations on the problems examined.

Tillman and Cochran [23] also extended the work of Clarke and Wright. Their method chooses the pair of nodes with the best savings such that the second best feasible pair may be chosen. This manner of choosing the best two feasible pairs of nodes maximizes the savings over four nodes, not two.

Golden, et al. [14], proposed a heuristic that modifies the Clarke and Wright algorithm in three ways:

- 1) by using a 'route shape' parameter  $\gamma$  to calculate a new savings

$$S_{ij} = d_{oi} + d_{jo} - \gamma d_{ij} .$$

The value of  $\gamma$  is varied over some range and the best set of routes are selected;

- 2) by only calculating a savings between nodes close to each other; and
- 3) by storing the savings calculations in a "heap structure" to reduce comparisons and improve the speed of the algorithm.

Robbins [20] has shown it possible to generate better Clarke and Wright solutions using the second modification above. It should be stated, though, that this happens rarely and the quality of the solution will in general be worse than when all savings are calculated.

Christofides and Eilon [6] developed an algorithm that solves routing problems by a three-optimal tour method. The three-opt method, as it is called in the literature, begins with a set of random routes. The procedure continues by removing three arcs from a route and replacing with three different arcs. Figure 2, an example of three unconnected arcs, demonstrates eight ways they can be reconnected to form a route. When this is repeated for all combinations of three arcs, a three-optimal tour is obtained.

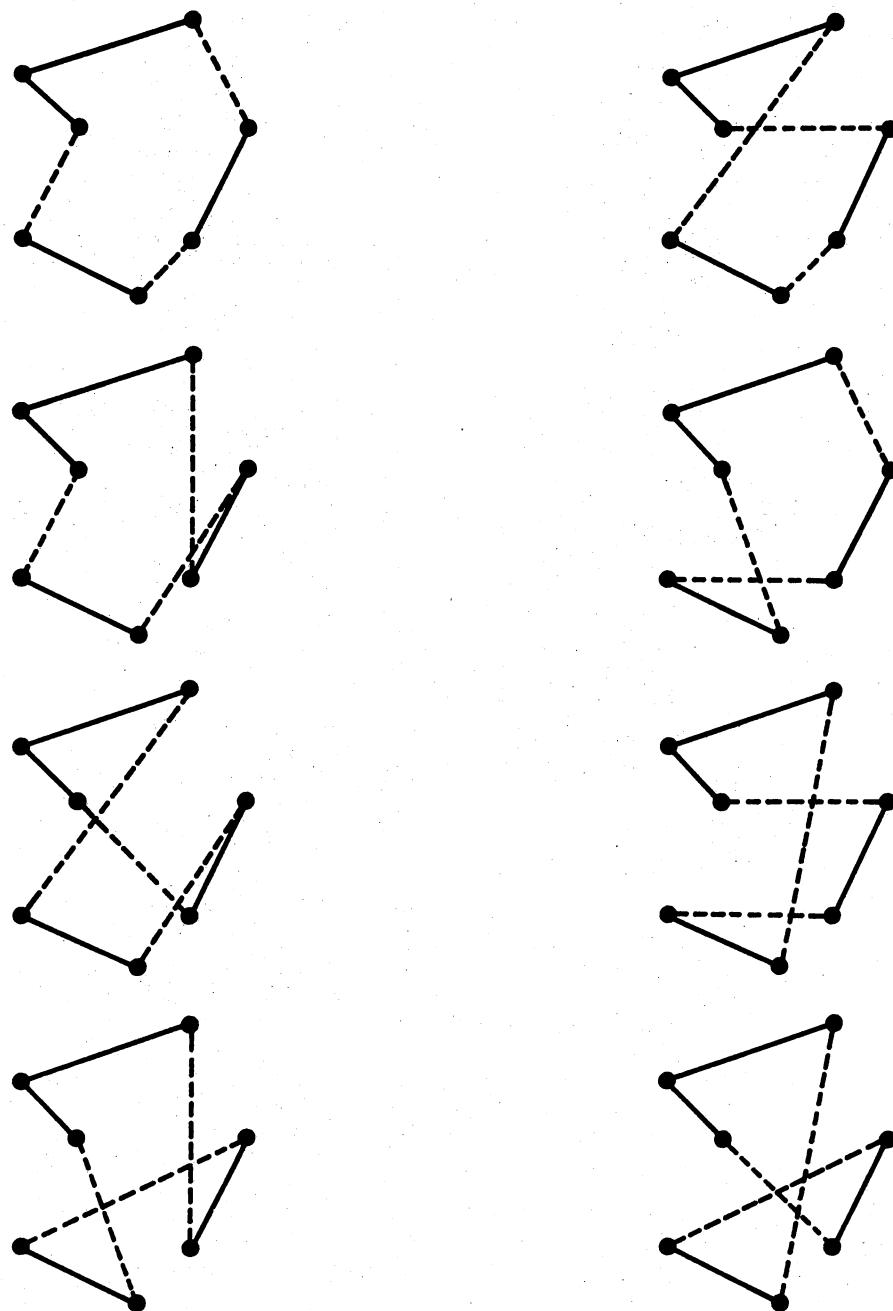


Figure 2. Eight Ways of Connecting Three Arcs

Gillett and Miller [13] recently proposed a procedure called the "sweep algorithm". The method involves labeling every node with its polar coordinates. The nodes are then numbered according to polar coordinate with the depot considered as location 1. A sweep is made such that nodes 2, 3, 4, ..., K are placed on the first route. Node K is the last node on the route that can be serviced without exceeding the carrier's capacity. This process continues until all nodes are on a route. A swapping routine then follows where stops are transferred between wedges if a reduction in distance is realized. A useful aspect of the algorithm is the time it takes to solve a problem. The authors report a linear increase in time with respect to the number of nodes if the number of nodes per route remains approximately constant.

Krolak, Felts, and Nelson [16] have developed a man-machine two phased procedure for routing problems. The procedure is very similar to that proposed by Krolak, Felts, and Marble [15] for the TSP. The first phase consists of a heuristic, called the "Truck-Route Generator", to aggregate nodes according to their location. Grouping sizes are kept small to create many clusters. Aggregates are combined where the total demand does not exceed the capacity of the largest vehicle. Results are probably not feasible but a TSP is solved for each group. Swaps between routes are then analyzed and made where possible. Finally, feasible one and two arc swaps are made within each route. The second phase of solution takes place at a cathode ray tube. The CRT displays the solution so an analyst can alter the routes in any way he desires as long as they remain feasible. This sort of interaction allows experience and intuitive knowledge to achieve better solutions. This,

in turn, makes the results more acceptable to an analyst as he plays a part in deriving the routes.

International Business Machines (IBM) [17] has a computer package called Vehicle Scheduling Program (VSP) that has been used to route vehicles. The program is primarily based on the Clarke and Wright procedure discussed earlier. Different factors can be minimized such as time, distance, or number of vehicles used. Where the designated factors cannot be minimized, the program tries to achieve a balance among them. In addition, VSP is well programmed and can handle over 1000 nodes.

#### Summary

Routing problems can be solved by many algorithms. Some procedures are exact while others are heuristic. Exact solution procedures generate optimal answers but are only practical for problems up to about 50 nodes. Large scale problems must be solved by heuristic techniques. Of the heuristics, the Clarke and Wright [7] method has been given the most attention. Gaskell [12], Tillman and Cochran [23], Golden [14], and IBM [17] have extended Clarke and Wright to produce procedures of their own. Other methods include those proposed by Gillett and Miller [13], Krolak, et al. [16], and Christofides and Eilon [6].

The next chapter presents the Clarke and Wright algorithm in detail. Also, an efficient program of the procedure is outlined.

## CHAPTER III

### THE CLARKE AND WRIGHT ALGORITHM

#### Introduction

This chapter discusses the Clarke and Wright algorithm [7] mentioned in Chapter II. The theory behind the algorithm and an example showing its use are presented along with the advantages and disadvantages. In addition, a computer program of the Clarke and Wright method is outlined.

#### The Clarke and Wright Algorithm

The Clarke and Wright routing procedure is based upon the calculation of a "saving" between every pair of nodes. The method initially assumes the existence of one vehicle to service each node. This is unrealistic as a vehicle usually can serve a great number of nodes. Clarke and Wright realized this and asked the question, "How much distance will be saved if one truck services two nodes instead of one?" As shown in Figures 3 and 4, the savings associated with pairs of two nodes can be calculated as:

$$S_{ij} = d_{oi} + d_{jo} - d_{ij} \quad \forall i \text{ and } j, i \neq j \quad (3.1)$$

where  $S_{ij}$  is the savings associated with pairing  $i$  and  $j$  on one route,  $d_{ij}$  is the distance from  $i$  to  $j$ , and  $o$  denotes the depot. "Distance" may be replaced by "money," "time," or any other scarce resource.

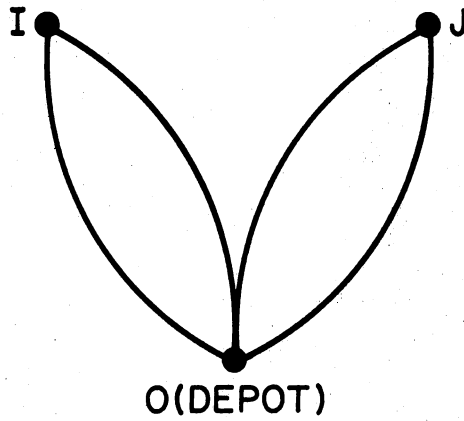


Figure 3. One Vehicle  
Servicing  
Each Node

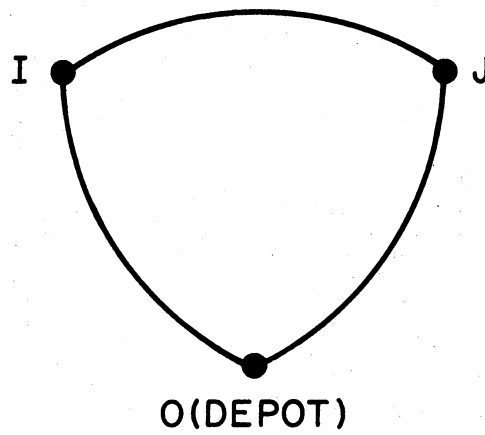


Figure 4. One Vehicle  
Servicing  
Two Nodes

Equation 3.1 is valid for symmetric or asymmetric distances. Since this research deals with symmetric distances, the asymmetric case will not be discussed further. For symmetric distances, the savings equation may be rewritten as:

$$S_{ij} = d_{oi} + d_{oj} - d_{ij}, \text{ or} \quad (3.2)$$

$$S_{ij} = d_{io} + d_{jo} - d_{ij} . \quad (3.3)$$

Once the  $S_{ij}$  are calculated for all pairs of nodes, the savings are ranked in non-increasing order. Going down the savings list, each pair of nodes is examined.\* For any given pair, an attempt is made to:

- 1) create a new route, or
- 2) add a node to the front or back of a route, or
- 3) join two routes to form one.

If none of the above can be accomplished, the nodes are discarded. This procedure continues until all pairs of nodes have been considered. An example is given below.

#### Example

Table I is a matrix of symmetric distances. Since the distances are symmetric, only half the matrix is needed. From the distance matrix the savings in Table II can be generated. For example, the savings for nodes 3-4 is:

$$\begin{aligned} S_{3,4} &= d_{0,3} + d_{4,0} - d_{3,4} \\ &= 9 + 13 - 6 \\ &= 16 \\ &= S_{4,3} \end{aligned}$$

---

\* Ties are broken by randomly selecting a pair.



TABLE I  
DISTANCE MATRIX

	j				
i	1	2	3	4	5
0	14	15	9	13	5
1		11	9	15	9
2			10	16	12
3				6	6
4					10

TABLE II  
SAVINGS MATRIX

	2	3	4	5
1	18	14	12	10
2		14	12	8
3			16	8
4				8

The savings of Table II are arranged in non-increasing order in Table III. The pair of nodes with the largest saving is 1-2. These stops become the first combination and form a route of 0-1-2-0.

TABLE III  
ORDERING OF SAVINGS

Stops	Savings	Stops	Savings
1-2	18	2-4	12
3-4	16	1-5	10
1-3	14	2-5	8
2-3	14	3-5	8
1-4	12	4-5	8

Next on the list is the combination 2-1. The addition of 2-1 to the first route creates a "subtour" which is not permitted. The next pair is 3-4. Since route 1 has no stop in common with this pair, a new route must be formed. Thus 0-3-4-0 forms route 2. Thus far, the routes appear as follows:

<u>Route #</u>	<u>Stops</u>
1	0-1-2-0
2	0-3-4-0

The next pair, 4-3, is dropped due to subtours. 1-3 is dropped since stop 1 has previously been traveled from. Likewise, 3-1 is dropped. If 1-2 were considered as 2-1, since the distance is symmetrical, the link 1-3 could be used to create one route from two: 0-2-1-3-4-0. This flexibility is difficult to incorporate into a program. Thus, a rule for the program is established: Once an order of arcs is established, the sequence will not be reversed.

The next pair, 2-3 allows two routes to become one: 0-1-2-3-4-0. Continuing down the list, all pairs are dropped except 5-1. Stop 5 is appended to the front of the route. Thus, after adding the depot, the path becomes 0-5-1-2-3-4-0 with a distance of 54 units. The location of nodes is shown in Figure 5 with the tour in Figure 6. Table I contains rectilinear distances while Figure 6 shows the route with directed arcs.

#### Advantages and Disadvantages

The Clarke and Wright algorithm has been very popular as indicated in Chapter II. Because of this exposure, many strengths and weaknesses have come to light. The following is a summary of the advantages and disadvantages of the procedure:

##### Advantages:

- 1) The procedure is simple to use.
- 2) A realistic constraint can be added easily.
- 3) The procedure provides a "good" starting solution which can be used as input to an improvement algorithm.

##### Disadvantages:

- 1) Once an arc or link is created it cannot be broken.
- 2) Results can be, but generally are not, optimal. In some cases where the constraints are tight, results are far from optimal.
- 3) A computer is required for most problems due to size (but this is true for most procedures).

The first advantage needs little elaboration. The procedure is straightforward and based on simple calculations. Realistic

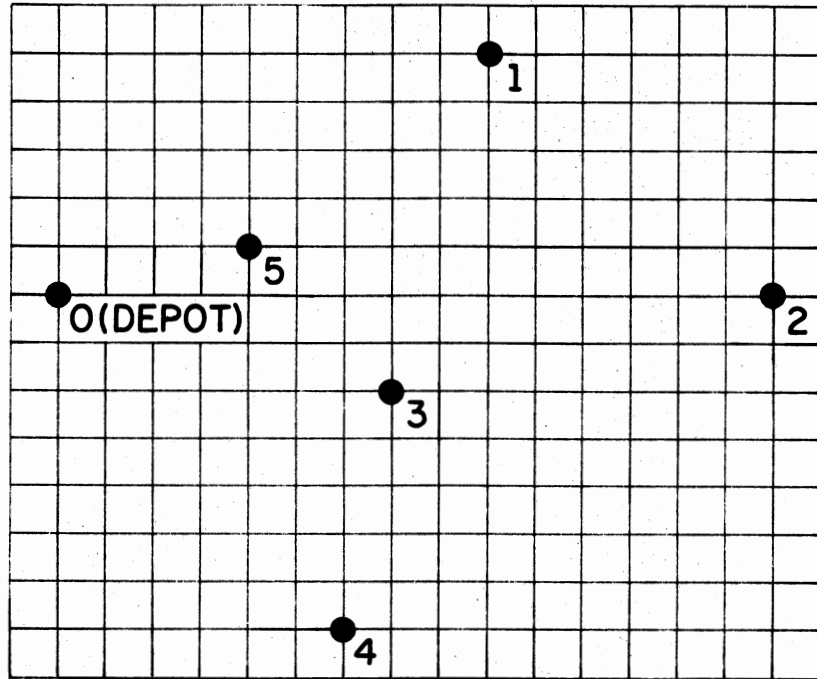


Figure 5. Node Locations

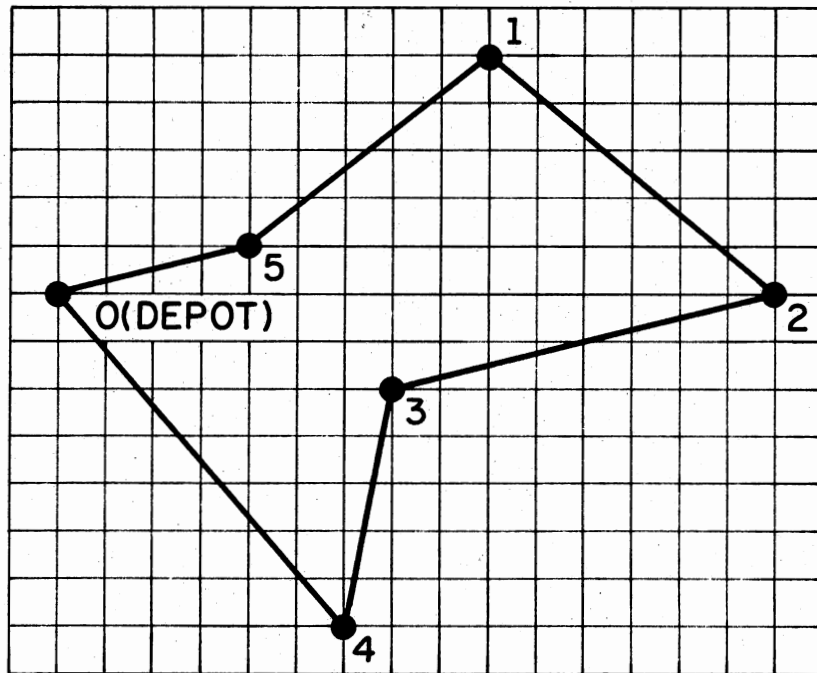


Figure 6. Clarke and Wright Route

constraints, such as maximum route length or required times of pickup, are easy to add. The ability to provide a good initial solution to a following improvement algorithm is crucial to this thesis. The Clarke and Wright algorithm generates a starting solution for use by the Lin two-opt procedure which is outlined in the next chapter.

The two-opt procedure attempts to improve an initial solution. If the starting solution is poor, the final solution may be poor. Likewise, a good starting solution will result in at least a good final solution. Primarily, it is this reason why the Clarke and Wright algorithm was chosen to build routes.

The disadvantages, while valid for the Clarke and Wright procedure, are partially remedied by the Lin two-opt algorithm. With respect to the first disadvantage, the two-opt procedure, which is a within-route swapping or perturbation routine, is used to break up links. Also, the third phase, man-machine interaction, allows further alteration of routes.

The second disadvantage must be kept in proper context. As previously discussed, optimal seeking algorithms use excessive computer time and normally are not practical. Thus, any attempt to find the optimal solution to a large scale routing problem is usually not feasible. As the third advantage states, this procedure provides a "good" starting solution for an improvement algorithm. The Lin two-opt procedure will, in general, improve the results enough to make the solution reasonable. The man-machine interactive phase may improve it even more.

## Programming Strategies

In this section the programming strategies incorporated in coding the Clarke-Wright algorithm are discussed. The program is written in FORTRAN with particular effort directed at reducing storage requirements and improving execution speed and efficiency. A flowchart of the program appears in Figure 7. Usage instructions are given in Appendix A while the program appears in Appendix B.

In order to efficiently carry out the Clarke-Wright procedure, it is necessary to retrieve specific information about nodes, distances, and routes without extensive searching. In order to accomplish this, the program incorporates a concept known as "linked list processing". Simply stated, linked list processing employs a set of pointers which provide rapid access to stored data. Use of these pointers generally results in less data manipulation and therefore faster execution times.

The program requires as input the number of nodes in the system, demand at each node, and a distance matrix. This distance matrix consists of the distances between all nodes in the system. Since these systems may be quite large, all distances and related route information are stored in half-word integer variables, i.e., INTEGER\*2. This reduces storage requirements by roughly one-half. Also by limiting computations to strictly integer variables, execution speed is greatly enhanced.

After the data has been read in, the savings calculations are performed. Next, these savings must be ordered. During this sorting process, an initial set of pointers is established. Through the remainder of the program, these and other pointers will be revised to

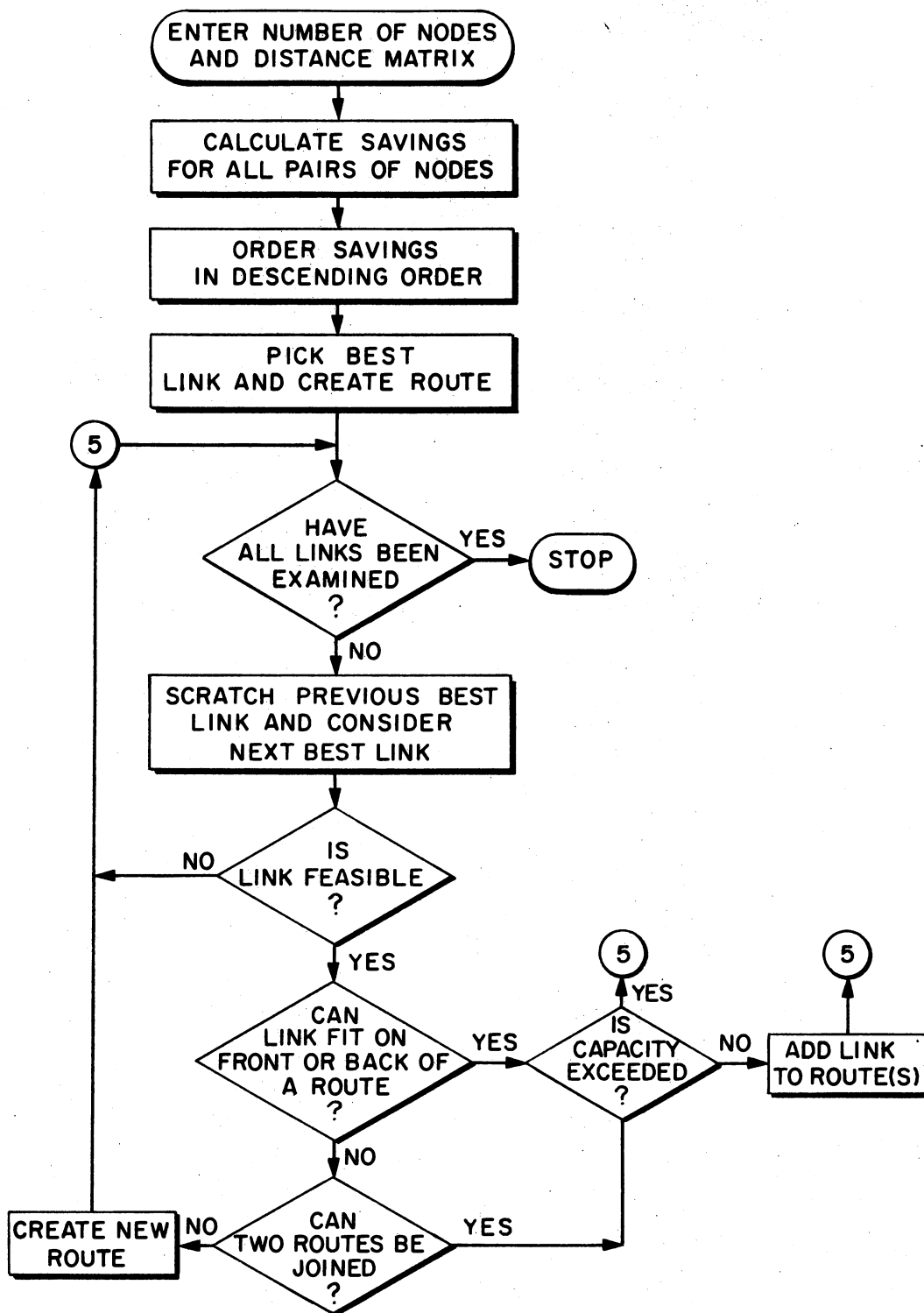


Figure 7. Flowchart of Clarke and Wright Algorithm

yield such information as:

- (a) Is a node on a route?
- (b) Which route is a node on?
- (c) Where on a route is the node located?

This information is then used to generate or improve potential routes. The updating of routes and pointers continues until the procedure iterates to completion.



## CHAPTER IV

### THE TWO-OPT ALGORITHM

#### Introduction

As originally conceived, the Lin two-opt algorithm [18] attempts to improve a traveling salesman tour. Given a route through  $N$  nodes, two arcs are removed from the circuit and replaced with two different links. If a reduction in total distance is realized by this swapping of arcs, the new links are retained and the tour is reexamined. Otherwise, the proposed links are discarded and examination of arcs continues. The procedure stops when a tour has been totally examined and no swaps have been made.

The two-opt procedure can be easily extended to routing problems. A feasible solution to a routing problem contains  $M$  routes, where  $M > 1$ . The case where  $M = 1$  is a traveling salesman problem. By considering each route independently of the others, a two-opt procedure can be used for possible improvement. Figure 8 shows two routes which, when considered independently, cannot be improved further by the two-opt method. If a swap of arcs between routes is considered though (7-0 and 9-4 to replace 7-4 and 0-9), the total distance might be reduced. Figure 9 shows the results of making the proposed swap. Unfortunately, between-route swapping is difficult to program. Thus, it is not included in this thesis. Instead, the man-machine interaction phase is incorporated which allows for analysis of between-route swapping.

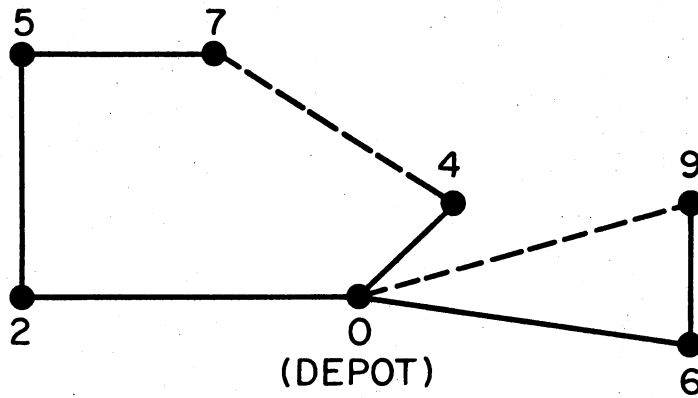


Figure 8. Two Independent Two-Opt Routes

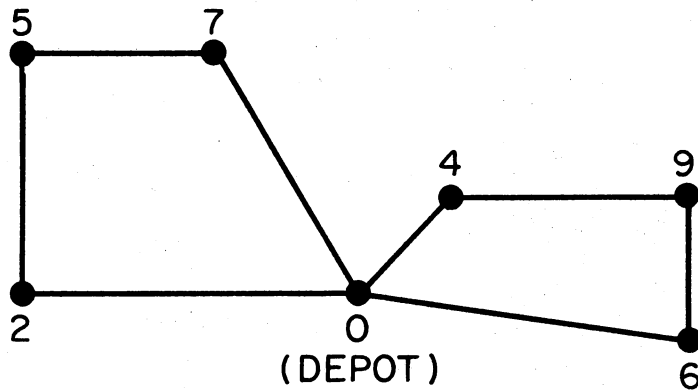


Figure 9. Swapping Arcs Between Routes

Figure 10 shows two routes, one of which can be improved with the two-opt procedure. Route A is two-optimal (when considered independently) while route B can be improved since it has two links that intersect, 6-4 and 3-0. Lin [18] states that any route containing an intersection of two or more links cannot be two-optimal. By swapping links 6-4 and 3-0 (the dashed lines, Figure 10) for 6-3 and 4-0 (Figure 11), route B becomes independently two-optimal.

### Two-Opt Requirements

The two-opt procedure, like most other improvement algorithms, starts with an initial feasible solution, which in this case is the Clarke and Wright algorithm. In addition to the starting solution, a distance matrix must also be provided so the feasibility of swaps can be determined.

A swap of links can be made two ways within a route, but only one of the swaps is realistically possible as shown in Figure 12. The feasible swap in the middle of Figure 12 may result in a better route while the infeasible swap at the bottom of Figure 12 results in subtours, which are not allowed. Assuming I-J and K-L are the arcs considered for replacement, the initial route is L-N-M-I-J-T-S-K-L while the possible replacement route is L-N-M-I-K-S-T-J-L. These two routes are identical except for the links between I and L which includes those being considered. (The links I-J and K-L are being considered for replacement by I-K and J-L.) The remaining links in the replacement route are K-S-T-J. Upon examination of the initial route, the links J-T-S-K are found, which are the reverse of those in the replacement route. Since distances are symmetric, the links from K to J are the same

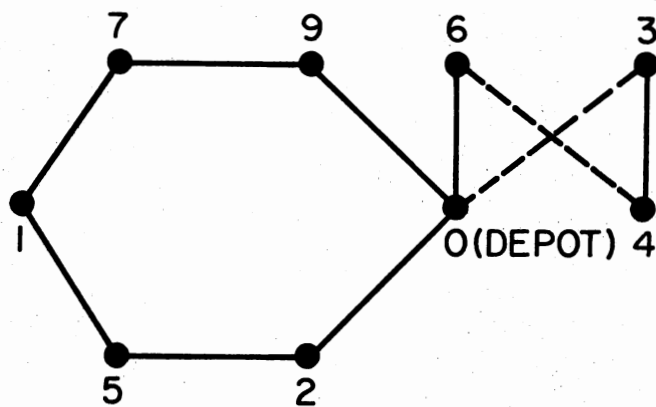


Figure 10. System of Two Routes

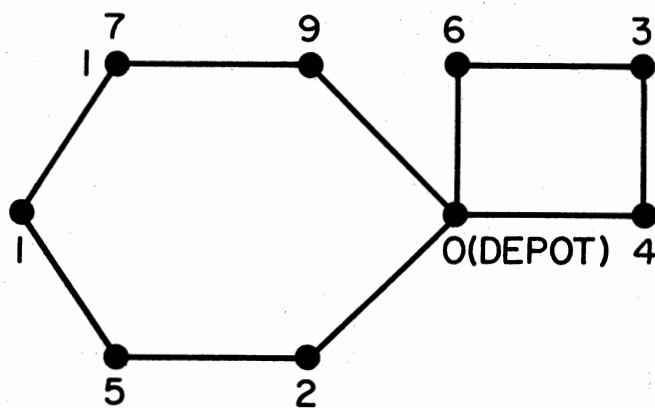
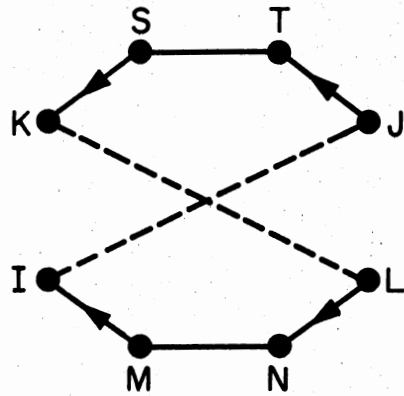
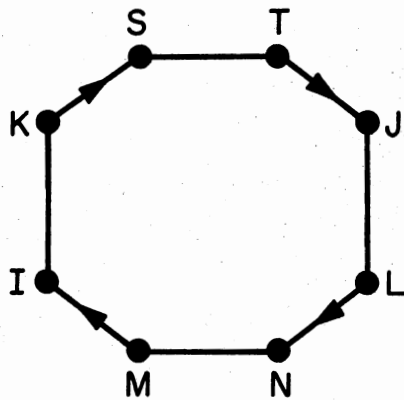


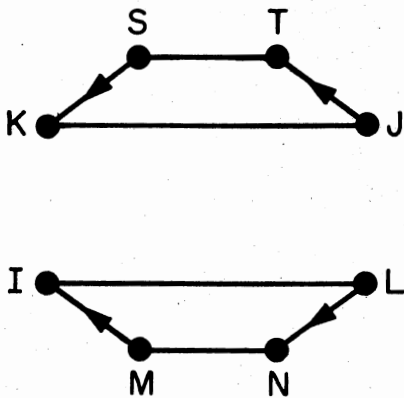
Figure 11. Replacement of Two Arcs on One of Two Routes



INITIAL ROUTE: L-N-M-I-J-T-S-K-L



FEASIBLE SWAP: L-N-M-I-K-S-T-J-L



INFEASIBLE SWAP: L-N-M-I-L AND J-K-S-T-J

Figure 12. Initial Route and Two Swaps

distance as from J to K. Thus, the links from K to J need not be considered when examining a swap. This leaves the arcs I-J and K-L as the basis for a swap. Since I-K and J-L are being considered to take the place of I-J and K-L, the following equation must be satisfied:

$$d_{I,K} + d_{J,L} < d_{I,J} + d_{K,L} \quad (4.1)$$

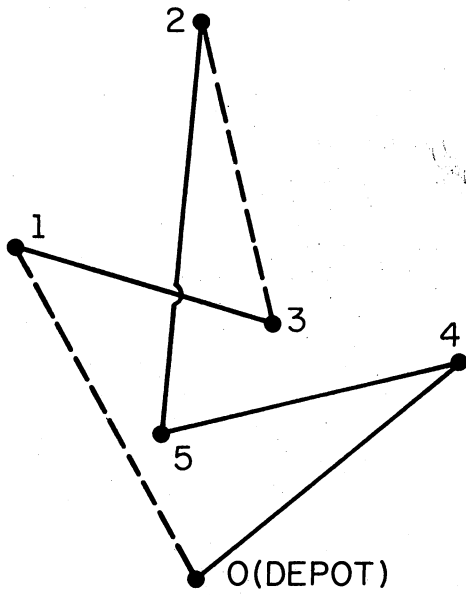
If equation 4.1 is satisfied, a swapping of arcs takes place. Otherwise, the route remains in its present form. Swapping arcs results in just two new arcs. All other links retain their initial sequence or are the reverse of their original order. It should be pointed out that the entire examination process could be reversed since the distances are symmetric; but the end result would be identical.

#### Example

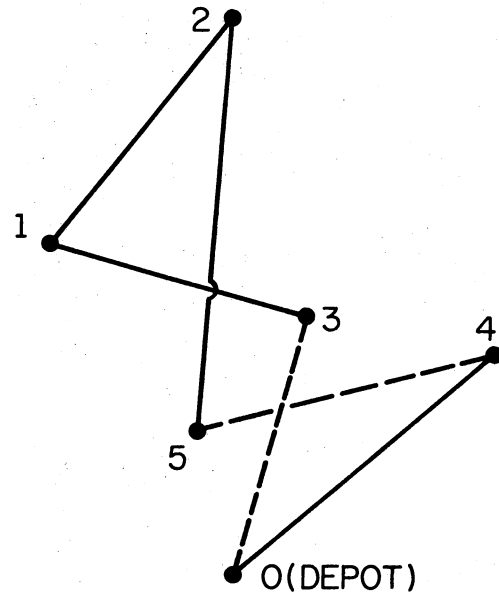
Figure 13 is a diagram showing how a route may be improved by the two-opt method. The initial solution (iteration 1) is a route of 68 miles. In iteration 1, the arcs 0-1 and 3-2 are considered for removal. The replacement links are 0-3 and 1-2. Since a reduction in distance can be made (from 68 to 63 miles) the swap is made in iteration 2. This swapping of arcs continues, where swapped arcs are dashed lines, in iterations 3 and 4 until the minimum distance of 54 miles is reached.

#### Programming Strategies

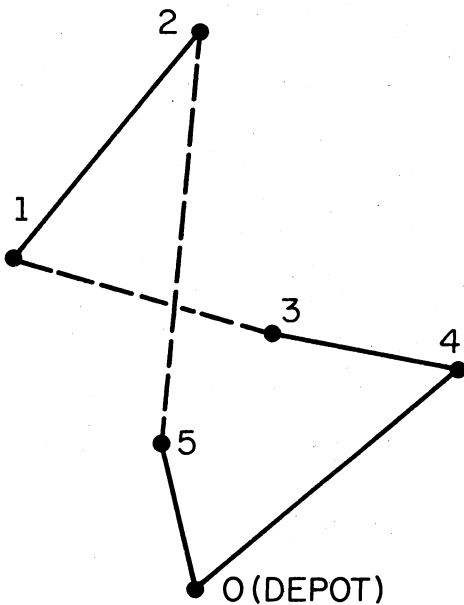
Since the first phase of the program is a Clarke and Wright procedure, a good initial solution is obtained. This is then used to feed into the two-opt within-route procedure, a flowchart of which is shown in Figure 14. The two-opt algorithm makes use of the pointers established in the Clarke and Wright procedure. Whenever a swap is deemed feasible, the pointers are rearranged and the route is reexamined starting from the beginning. This reexamination is necessary because



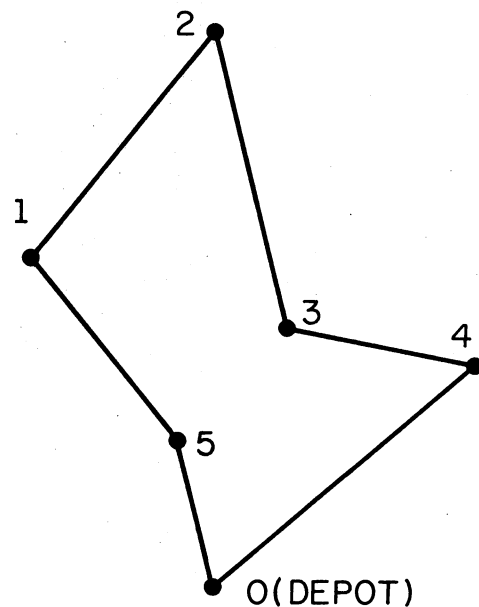
ITERATION 1  
68 MILES



ITERATION 2  
63 MILES



ITERATION 3  
55 MILES



ITERATION 4  
54 MILES  
NO FURTHER IMPROVEMENT

Figure 13. Sequence of Two-Optimal Changes Within One Tour

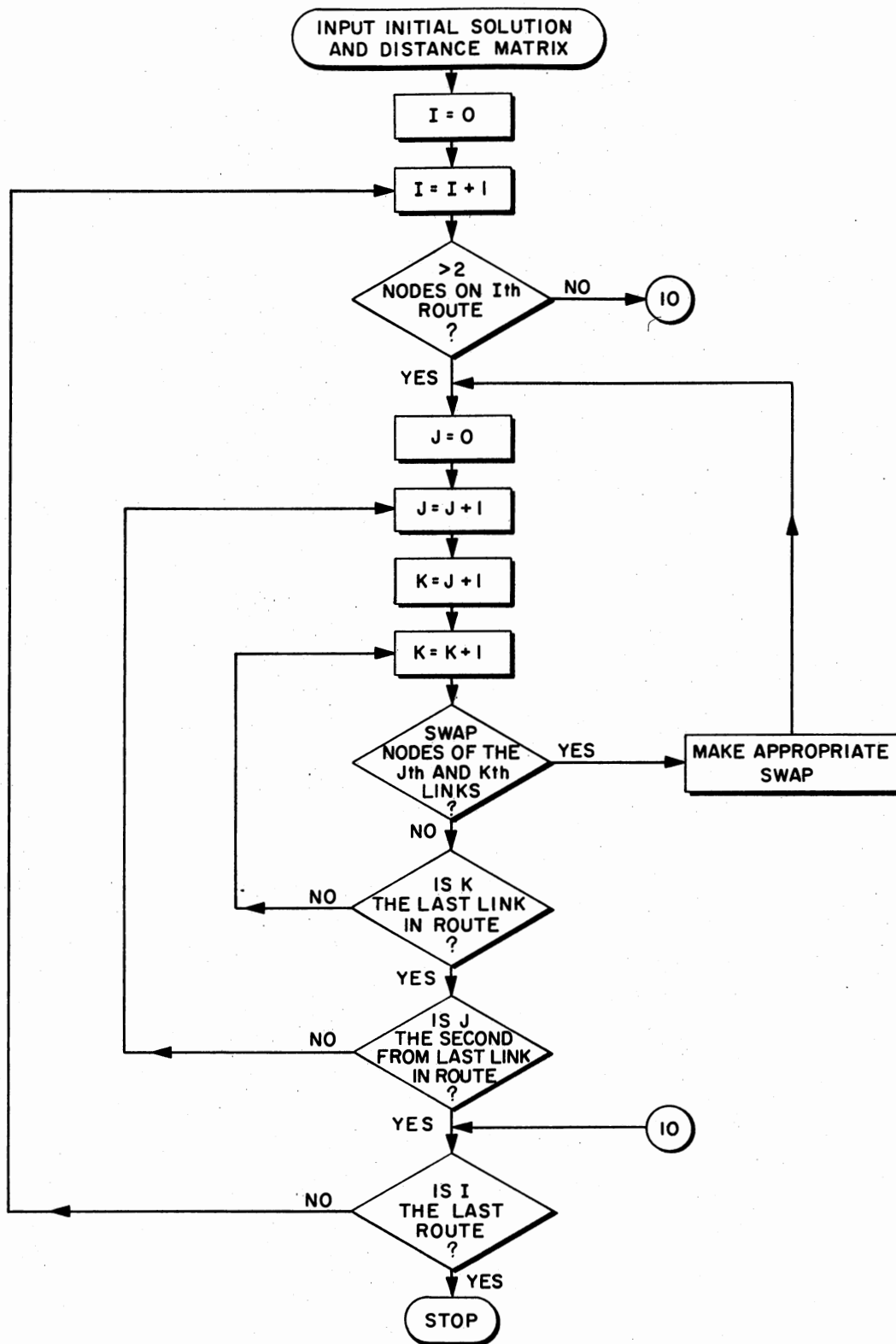


Figure 14. Flowchart of Lin Two-Opt Algorithm



new links are introduced into the route which may make more swaps feasible. The algorithm ends when a route is examined and no swaps are made.

## CHAPTER V

### COMPUTATIONAL RESULTS

#### Introduction

This chapter presents some computational experience of the program developed in this thesis. Problems solved by Christofides and Eilon [6], Gaskell [12], and Gillett and Miller [13] are examined.

#### Background of Problems

Christofides and Eilon [6] solve ten vehicle routing problems with their three-opt procedure (and other procedures) and compare the results to those obtained by one of Gaskell's savings algorithms [12]. In most cases, the three-opt procedure produces better results but takes considerably more time for solution. Of the ten problems, seven have less than fifty nodes and are not examined here. The three remaining problems (50, 75, and 100 nodes) are solved by Gillett and Miller [13] along with five others. These eight problems are presented in Table IV. Problems 3, 4, and 5 are the same as 2 except the maximum load is altered. Likewise, problem 7 is the same as 6 except for the load constraint. Solution results are presented in Table V.

TABLE IV  
LIST OF PROBLEMS

Problem Number	Author	Number <sup>a</sup> of Nodes	Maximum Load
1	Christofides and Eilon [6]	50	160
2	Gillett and Miller [13]	75	100
3	Christofides and Eilon [6]	75	140
4	Gillett and Miller [13]	75	180
5	Gillett and Miller [13]	75	220
6	Gillett and Miller [13]	100	112
7	Christofides and Eilon [6]	100	200
8	Gillett and Miller [13]	249	500

<sup>a</sup>Excludes depot.

TABLE V  
COMPARISON OF ALGORITHMS

Problem Number	Three-Opt			Gaskell Savings			Sweep Algorithm				(Proposed Algorithm) Clarke-Wright/Lin			
	Sol.	Rts.	Min.	Sol.	Rts.	Min.	Sol.	Rts.	Min.	Avg. No. nodes/route	Sol.	Rts.	Min.	Avg. No. nodes/route
1	556	5	2.0	585	6	.6	546	5	2.00	10.0	580	6	.07	8.3
2							1127	15	.68	5.0	1074	14	.12	5.4
3	876	10	4.0	900	10	1.3	865	10	1.23	7.5	892	11	.12	6.8
4							754	8	2.23	9.4	790	8	.12	9.4
5							715	7	3.68	10.7	728	7	.12	10.7
6							1170	14	1.83	7.1	1162	14	.20	7.1
7	863	8	10.0	887	8	2.5	862	8	6.00	12.5	897	8	.20	12.5
8							5794	25	9.70	10.0	5457	26	2.74	9.6

### Examination of Results

Table V shows the results of four different solution procedures on eight problems. The four procedures are:

- (a) Three-Opt Procedure (results available on only three problems),
- (b) Gaskell Savings Procedure (results available on only three problems),
- (c) Gillett and Miller Sweep Algorithm,
- (d) Clarke and Wright-Lin Algorithm (procedure developed in this thesis).

The sweep algorithm outperforms the three-opt and Gaskell savings routine for problems where comparisons can be made (1, 3, and 7). Computer times are difficult to contrast since each algorithm was programmed on a different computer. Thus, one might conjecture (unscientifically) that the sweep algorithm is at least as good on the eight problems as the other two procedures.

For that reason, the Clarke and Wright-Lin program will be compared to the sweep algorithm.

The Clarke and Wright-Lin program was run on an IBM 360-65 at Oklahoma State University. Gillett and Miller ran the sweep program on an IBM 360-67. The only significant advantage the IBM 360-67 has over the IBM 360-65 is the ability to handle more programs. Storage capability and execution speed are virtually the same. Thus, for practical purposes, the two programs were executed on the same computer.

As shown in Table V, the sweep algorithm produces better solutions for all problems except 2, 6, and 8. The maximum difference in solutions is 5.8% (problem 8) while the sweep algorithm produces

results averaging 1% better than the Clarke and Wright-Lin procedure. Thus, on the eight problems examined, these two methods perform almost equally as far as solution is concerned.

Another factor of interest is the computer time of the two programs. Computer time is defined as the execution time of the program. Gillett and Miller report a linear increase in computer time with an increase in the number of nodes if the number of nodes per route remains relatively constant. Also, for any given route, the computer time increases as the number of nodes per route increases. This can be seen in Table V for the 75 node problem. Computer time ranges from .68 minutes to 3.68 minutes while the average number of nodes per route varies from 5 to 10.7. Thus, computer time may become excessive for a problem with a large number of nodes per route.

Computer time for the Clarke and Wright-Lin procedure is independent of the number of nodes per route. Thus, although the average number of nodes per route increases from 5.4 to 10.7 (Table V), the time for solution of the four 75 node problems remains constant (.12 minutes). Additionally, computer time for the Clarke and Wright-Lin procedure is significantly less than the sweep algorithm for each of the eight problems considered. As in most heuristic solutions though, computer time increases exponentially with the number of nodes. Table V shows computer time increasing from .07 minutes for 50 nodes to 2.74 minutes for a 249 node problem. For some number of nodes, the execution time for the two procedures should be approximately equal. These results show that this probably occurs somewhere around 350 nodes. Thus, for problems containing less than 350 nodes, the Clarke and Wright-Lin algorithm is probably best to use, as computer time will be

smaller than for the sweep algorithm. Past 350 nodes, the sweep algorithm is probably best to use.

### Statistical Analysis

For a statistical comparison of the proposed algorithm and the sweep procedure, it is probably best to turn to non-parametric tests. The sign test is used to test two hypotheses. The first, which is based on solutions generated by the eight problems, can be stated as:

$$H_0: P(X_1 > X_2) = P(X_1 < X_2) = .5$$

$$H_1: P(X_2 < X_1) > .5$$

where  $X_1$  and  $X_2$  are the solutions of the Clarke and Wright-Lin program and sweep algorithm respectively. This is a one tailed test and the critical test value, for a 5% significance level, is 3.84. Table VI shows three positive and five negative signs. The experiment's statistic can be calculated as:

$$\begin{aligned} \chi^2 &= \frac{(3 - 4)^2}{4} + \frac{(5 - 4)^2}{4} \\ &= .5 \end{aligned}$$

Since .5 is less than 3.84, the null hypothesis,  $H_0$ , cannot be rejected. Thus, there is no statistical difference between the solutions of the Clarke and Wright-Lin algorithm and the sweep algorithm.

The second hypothesis can be stated as:

$$H_0: P(X_1 > X_2) = P(X_1 < X_2) = .5$$

$$H_1: P(X_1 < X_2) > .5$$

where  $X_1$  and  $X_2$  are the execution times of the Clarke and Wright-Lin program and sweep algorithm respectively. This is also a one tailed

TABLE VI  
SOLUTION RESULTS AND DIFFERENCES FOR SIGN TEST

Problem Number	Sweep Algorithm	Clarke-Wright/ Lin	Sign of Difference
1	546	580	-
2	1127	1074	+
3	865	892	-
4	754	790	-
5	715	728	-
6	1170	1162	+
7	862	897	-
8	5794	5457	+



test with a critical test value of 3.84 for a 5% significance level. Since Table VII shows eight positive signs, the following statistic can be calculated:

$$\begin{aligned}\chi^2 &= \frac{(8 - 4)^2}{4} + \frac{(0 - 4)^2}{4} \\ &= 8.0\end{aligned}$$

Since 8.0 is greater than 3.84, the null hypothesis can be rejected. Thus, it seems safe to suggest the Clarke and Wright-Lin program is at least as fast in execution time as the sweep algorithm.

#### Additional Computational Experience

Computational experience using the program proposed in this thesis has also been gained by solving six randomly generated problems. Node coordinates and demands are given in Appendix C and the solution results are presented in Table VIII. Note that as the number of nodes increases, the execution time and required storage increase exponentially. Figure 15 is a graph of the execution time versus number of nodes while Figure 16 shows the graph of storage requirements versus number of nodes. In both cases, the curvature is the classical exponential curve.

TABLE VII  
EXECUTION TIMES AND DIFFERENCES FOR SIGN TEST

Problem Number	Sweep Algorithm	Clarke-Wright/ Lin	Sign of Difference
1	2.00	.07	+
2	.68	.12	+
3	1.23	.12	+
4	2.23	.12	+
5	3.68	.12	+
6	1.83	.20	+
7	6.00	.20	+
8	9.70	2.74	+

TABLE VIII  
SOLUTIONS TO RANDOMLY GENERATED PROBLEMS

Problem Number	Nodes	Solution	Number of Routes	Execution Time (seconds)	Memory Requirements (Bytes)
1	10	275.5	2	2.27	30,548
2	30	612.0	4	2.65	36,160
3	50	1011.4	8	4.12	46,160
4	70	1147.6	9	6.57	60,164
5	90	1495.6	13	11.05	80,104
6	150	2338.5	20	36.69	172,254

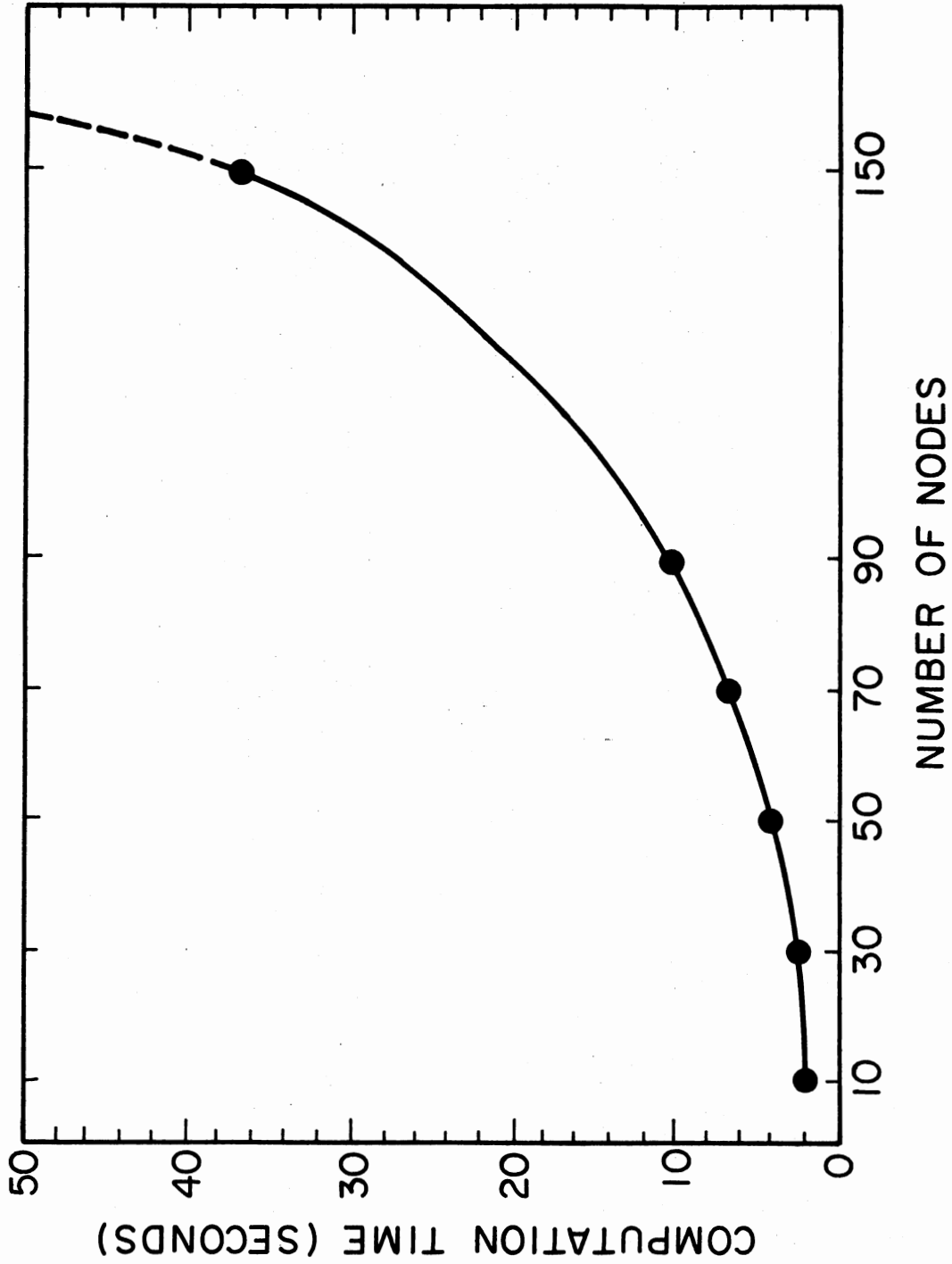


Figure 15. Computation Time Versus Problem Size

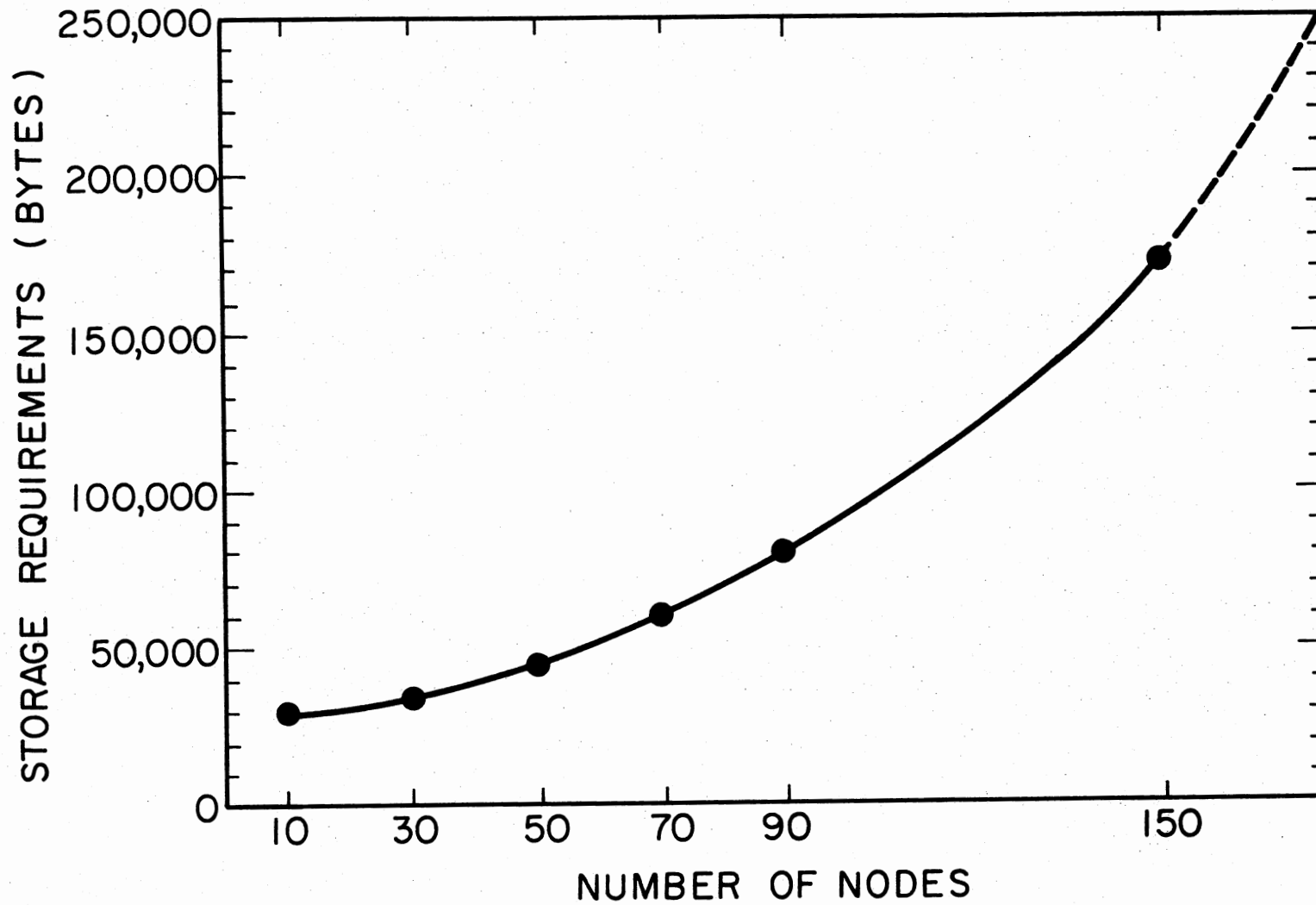


Figure 16. Storage Requirements Versus Problem Size

## CHAPTER VI

### MAN-MACHINE INTERACTION

#### Description

Solution of a large scale routing problem generally requires the use of a computer. Information is fed in, an algorithm executed, and final results are printed. In other words, an analyst gathers the data and the computer does the work. The analyst, by examining the computer solution, may be able to recommend changes that result in a reduction of distance since large scale computer solutions usually are not optimal. This implementation of changes in a discourse between analyst and computer is called "man-machine interaction". In this interaction, an analyst (man) examines the results of a computer (machine) and tries to create a better solution. This is then given to the computer which tries to improve the solution further. This interaction can be continued as long as desired. This chapter discusses the reasons for using an interactive phase and how it occurs.

#### Reasons For Use

This third phase of the computer program allows an analyst to interact with the computer to improve the first generated solution. After examination of the results provided by the Clarke and Wright-Lin phases, changes may be desired. For example, a route may be too long

or a vehicle may not serve enough nodes. These are only two of the many reasons a dispatcher may wish to alter the computer's results. Usually, for example, it is human nature for a dispatcher to have more faith in a solution he helps derive than in the results of a machine. Also, a dispatcher is usually very familiar with his territory as he may have spent years learning his trade and gaining experience. For example, he may know a certain road is dangerous in high water and should be avoided or a bridge may be overloaded with a full bus but not an empty one. The interaction phase allows this experience to be put to use. Also, many times a dispatcher may be interested in experimenting with his routes in the hope of finding a better solution. Experimentation can be done easily through the interaction phase. The effects of adding or deleting nodes, adding additional routes, or transferring nodes within the system can be realized very quickly and at low cost. With the use of a computer's speed and memory, many solutions can be generated with the best result chosen for use. Appendix D outlines the steps to be taken when altering the solution generated by the Clarke and Wright-Lin phases.

#### Usage

The first two phases of the program provide a solution to the routing problem. The Clarke and Wright algorithm (first phase) generates a solution and the Lin procedure (second phase) attempts to improve it. The interaction phase is also used for route improvement purposes. An analyst first receives the Clarke and Wright-Lin solution and examines it. If any changes are desired, the alterations are made and punched on data cards which are fed back into the computer. The

computer then skips the Clarke and Wright phase and attempts to improve the inputted solution using the within-route two-opt procedure. Results are again printed and the analyst decides what changes, if any, need to be made. This process continues until the analyst is satisfied with the results provided by the computer.

### Real World Implementation

Recently, the program of this thesis was used to find new routes for the school district of Yale, Oklahoma. Figure 17 is a picture of the school district with the location of nodes. Presently, six buses, each with a fifty-five student capacity, travel 166 miles to pick up two hundred nineteen children. Since the location and demand of nodes (school bus stops) is known, the only data needed is the distance matrix. Using a map and roughly forty man-hours, the matrix was generated and coded on data cards. After coding the other necessary data, the program was ready for use. The initial result, provided by the Clarke and Wright-Lin phases and shown in Figure 18, yielded a total distance of 148.5 miles and five buses. Every route is reasonable except route one which is too long (52.5 miles). Thus, the solution is deemed impractical. Were an interaction phase not available, the procedure would be a failure for this problem or at least manual changes would be necessary. Using the interaction phase, an examination of routes one and two (Figure 18) reveals some unnecessary overlapping near nodes 72 and 73. Keeping this in mind and the fact that route one must somehow be shortened, a new solution can be proposed. Route one can be broken into two routes. The first route consists of nodes 67, 66, 64, ..., 76, 74 in addition to 73 and 72.

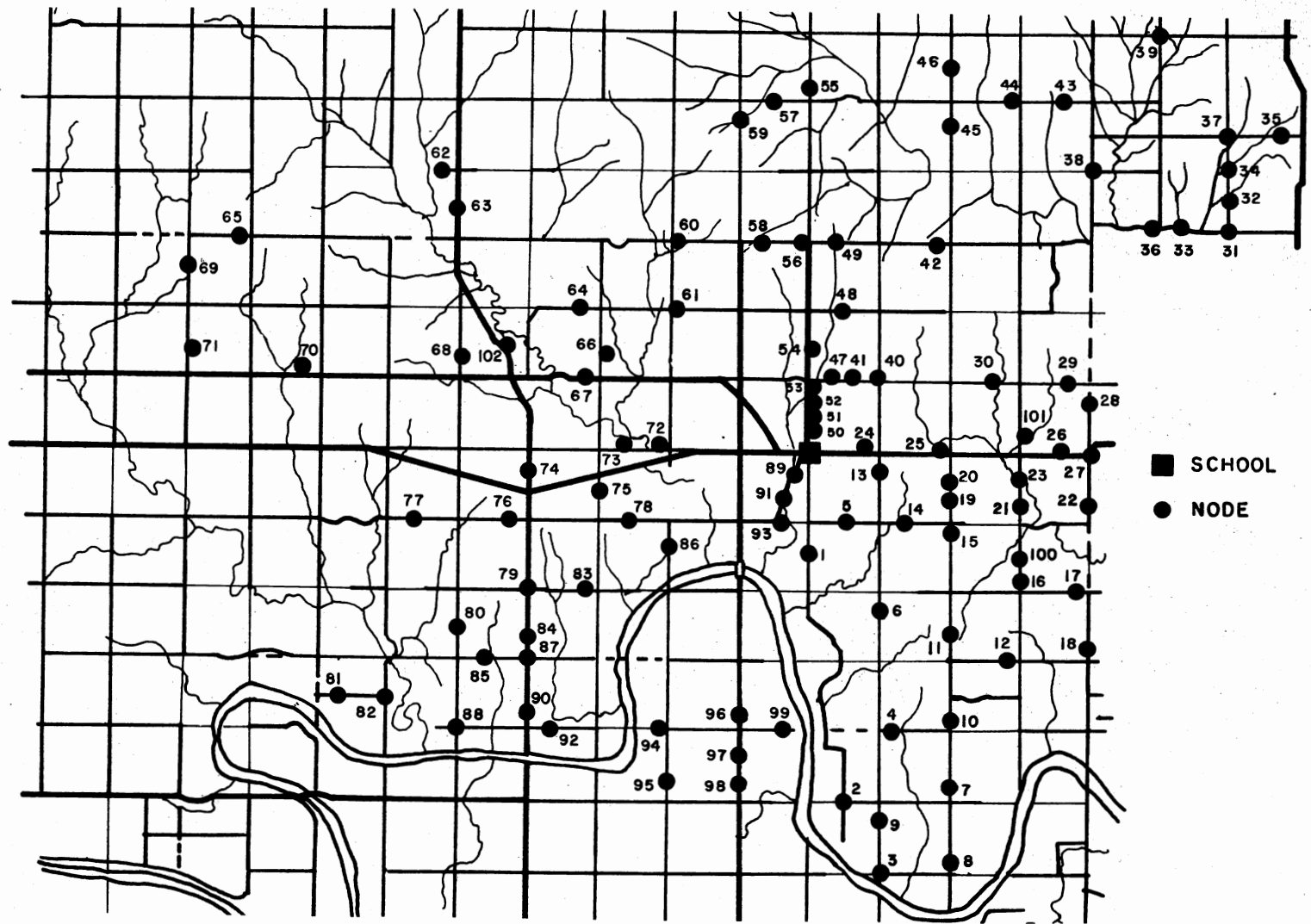


Figure 17. Yale School District



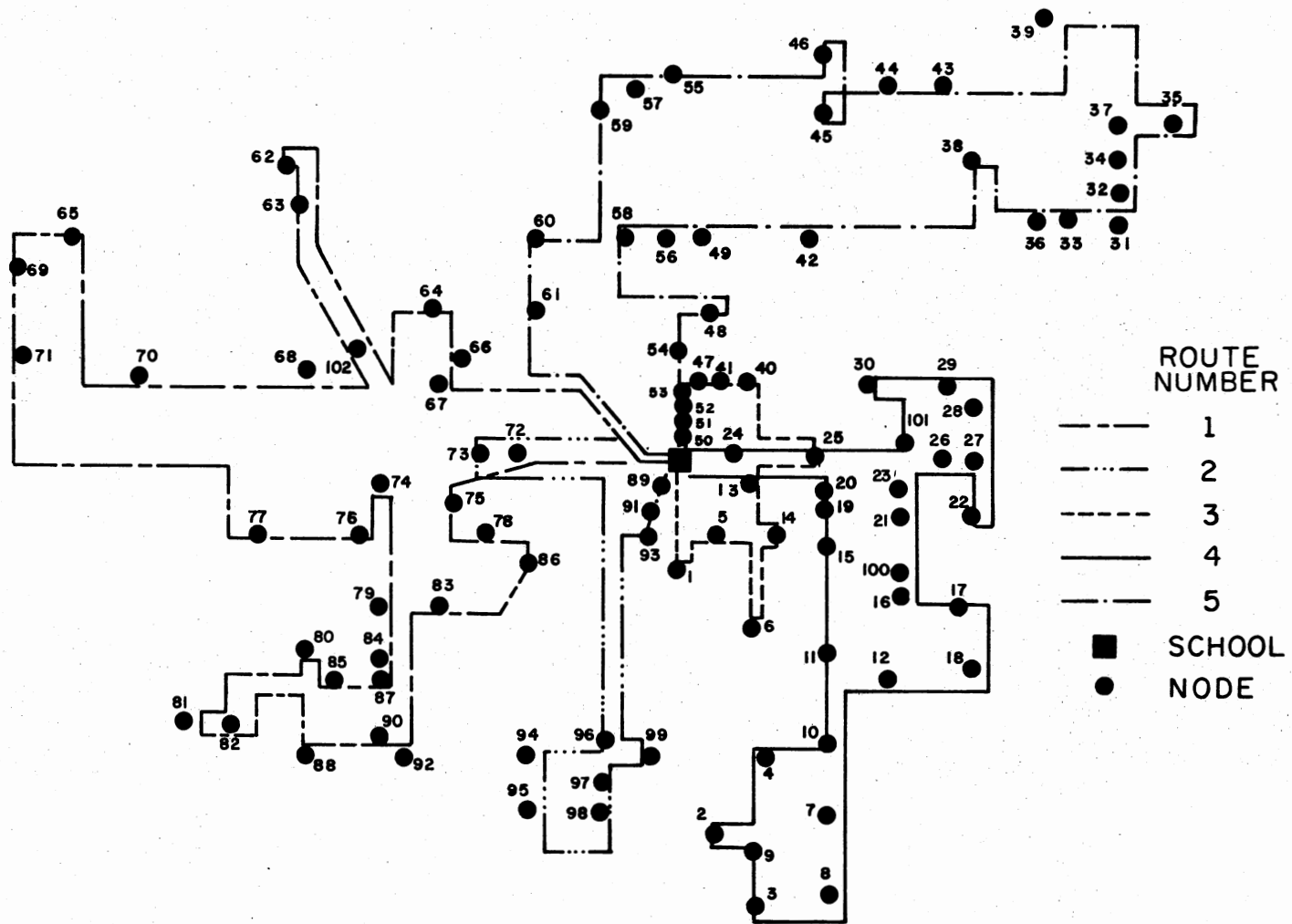


Figure 18. Initial Clarke-Wright/Lin Solution

Nodes 73 and 72 formally were part of old route two. The second new route is composed of nodes 89, 91, 93, ..., 79 and 75. These changes are fed into the computer and reapplication of the Lin procedure yields another solution. This revised solution is shown in Figure 19 with a distance of 152.5 miles and six buses. Even though an extra route is added to the system, the total distance increases by only four miles and the solution is acceptable. This is a good example of an analyst recognizing a constraint of the system and altering the solution accordingly.

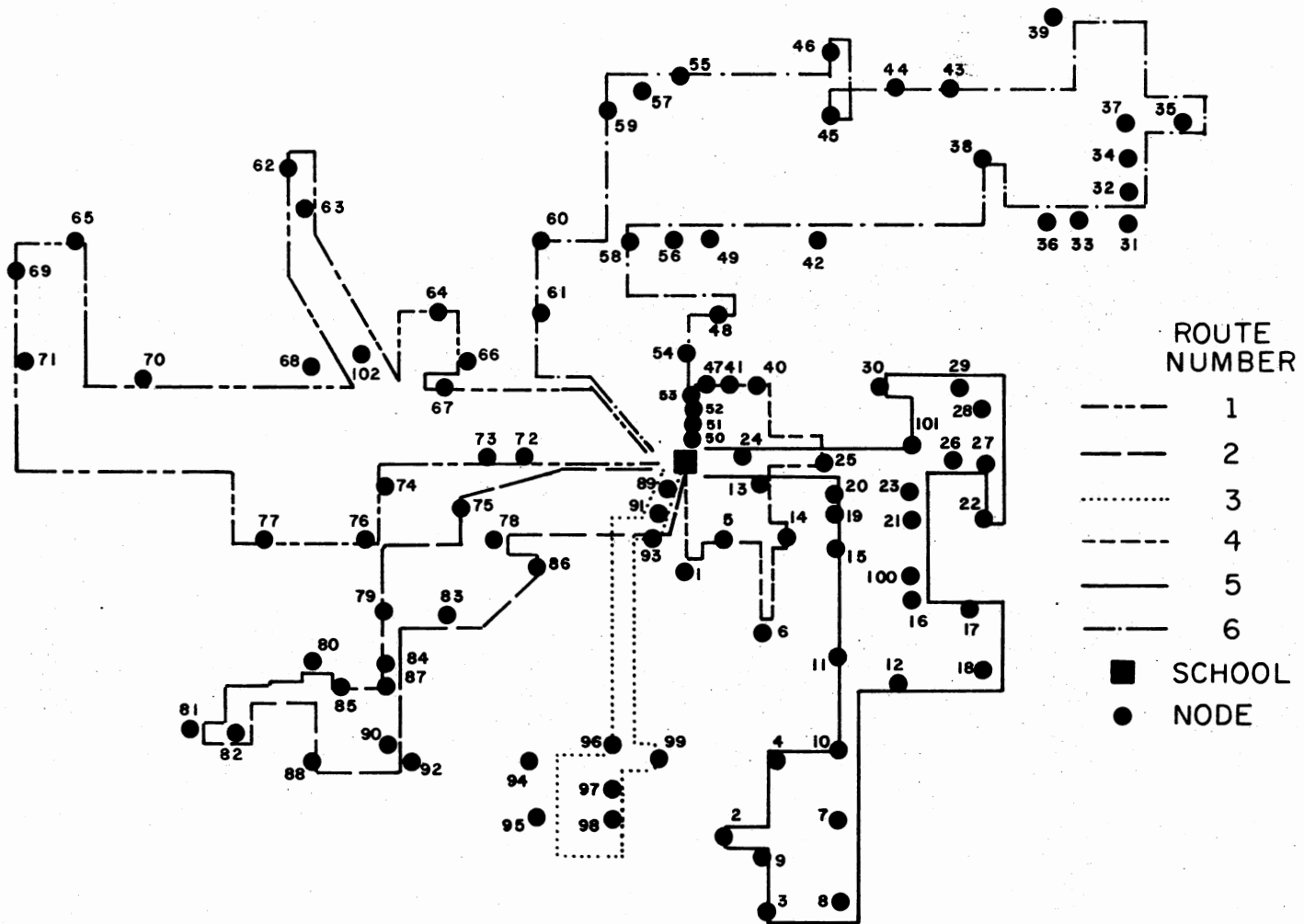


Figure 19. Interaction Solution

## CHAPTER VII

### CONCLUSIONS AND FUTURE RESEARCH

#### Conclusions

The first six chapters of this thesis present a three phase procedure called the Clarke and Wright-Lin Interaction Program for solution of routing problems. Problems presented in papers by Gillett and Miller [13] and Christofides and Eilon [6] have been solved with the proposed algorithm. Results presented in Chapter V show the proposed procedure to be competitive with the Gillett-Miller sweep algorithm as far as quality of solution is concerned. Both perform well on the same problems and no statistical difference can be determined for a significance level of 5%. The program developed in this research, however, has been shown, statistically, to be faster than the sweep algorithm for the problems solved. This should be taken in proper context as the proposed algorithm was intended to be programmed efficiently while the sweep algorithm may or may not have been.

Chapter VI discusses the third phase of the proposed program, the interaction phase. This phase opens up many alternatives for an analyst. Probably one of the most important aspects is the addition of another resource to help solve routing problems: the analyst's experience. This is a virtually untapped resource that can be turned into an asset if properly used. The Clarke and Wright-Lin Interaction

Program is designed to assist an analyst in solving problems, not blindly force him to accept generated answers. The analyst is led in the direction of better solutions; but ultimately, the final results are only as good as the analyst. In summary, the qualities of the program proposed in this thesis can be stated as follows:

- (a) Produces good results,
- (b) Is programmed efficiently so computational requirements are small, and
- (c) Allows for use of the dispatcher's experience.

#### Future Research and Extensions

As discussed in Chapter II, Christofides and Eilon [6] have used a three-opt procedure to solve routing problems. Using a random set of routes as an initial solution, the three-opt method is used to generate the final routes. Unfortunately, the final results are a function of the initial solution and thus are usually not very good. An area of exploration is the use of the three-opt procedure for route improvement instead of route generation. For example, the sweep algorithm could be followed with a three-opt procedure. Actually, any algorithm that generates routes can be followed by the three-opt procedure. Also, a three-opt procedure could be used in conjunction with the Clarke and Wright algorithm.

Probably the most difficult aspect of using any routing algorithm is the generation of a distance matrix. Recall that a distance matrix contains the mileage or cost of traveling from node  $i$  to node  $j$  for all  $i$  and  $j$ . In a private conversation with the author, Dr. James K. Byers of the University of Missouri-Rolla has recommended the use of a

distance plotter to calculate distance matrices. Given a set of nodes and a distance scale (one inch equals one mile, for example), a matrix can be generated by tracing with a special pen each  $d_{ij}$ . Byers reports generating a one hundred node matrix in roughly three man hours. As stated in Chapter VI, the author spent forty man hours generating a one hundred and two node distance matrix. Clearly, Byers' procedure for creating a distance matrix warrants exploration.

As outlined in Appendix A, an analyst must number each node in a system before generating the distance matrix. It is probably best to use a map to do this. This means the map must be referred to each time a solution is generated. A more efficient means of analysis is to allow the analyst a visual display where results can be seen without referring to a map. This can be accomplished through the use of a Cathode Ray Tube (CRT) or a plotter routine. A CRT is a tube that displays results on an apparatus similar to a television set. The CRT also may contain a keyboard which allows an analyst to interact with the computer. Clearly, a CRT lends itself to maximum utilization of the analyst's time. Since results are displayed visually, reference to a map is not required. Also, possible node changes can be seen and implemented quickly. Thus, an area of investigation is the computer software and hardware requirements for use of the proposed algorithm on a CRT.

## BIBLIOGRAPHY

- (1) Ashour, S., and R. G. Parker. "A Schedule Algebra Approach to the Traveling Salesman Problem." (Unpublished working paper, Department of Industrial Engineering, Kansas State University, Kansas, 1973.)
- (2) Bellman, R. "Dynamic Programming Treatment of the Traveling Salesman Problem." J. Assoc. Comp. Mach., 9 (1962), pp. 61-63.
- (3) Bellmore, M., and J. C. Malone. "Pathology of Traveling-Salesman Subtour-Elimination Procedures." Operations Research, 19, 2 (March-April, 1971), pp. 278-307.
- (4) Bellmore, M., and G. L. Nemhauser. "The Traveling Salesman Problem: A Survey." Operations Research, 16, 3 (1968), pp. 538-588.
- (5) Bodin, L. "A Taxonomic Structure for Vehicle Routing and Scheduling Problems." Computers and Urban Society, 1 (1975), pp. 11-29.
- (6) Christofides, N., and S. Eilon. "An Algorithm for the Vehicle Dispatching Problem." Operational Research Quarterly, 20 (1969), p. 309.
- (7) Clarke, G., and J. Wright. "Scheduling of Vehicles from a Central Depot to a Number of Delivery Points." Operations Research, 12, 4 (1964), pp. 568-581.
- (8) Croes, G. A. "A Method for Solving Traveling Salesman Problems." Operations Research, 6 (1958), pp. 791-812.
- (9) Dantzig, R., and J. Ramser. "The Truck Dispatching Problem." Management Science, 6, 80 (1959), pp. 81-91.
- (10) Eastman, W. L. "A Solution to the Traveling Salesman Problem." (Presented at the American Summer Meeting of the Econometric Society, Cambridge, Massachusetts, August, 1958.)
- (11) Eilon, Samuel, S. D. T. Watson-Gandy, and Nicos Christofides. Distribution Management. London: Griffin, 1971.
- (12) Gaskell, T. "Bases for Vehicle Fleet Scheduling." Operational Research Quarterly, 18 (1967), p. 281.

- (13) Gillett, B., and L. Miller. "A Heuristic Algorithm for the Vehicle Dispatch Problem." Operations Research, 22 (1974), p. 340.
- (14) Golden, B., T. Magnanti, and H. Nguyen. "Implementing Vehicle Routing Algorithms." (Presented at the ORSA/TIMS meeting, Las Vegas, Nevada, November, 1975.)
- (15) Krolak, P., W. Felts, and G. Marble. "A Man-Machine Approach Toward Solving the Traveling Salesman Problem." CACM, 14 (1971), pp. 327-334.
- (16) Krolak, P., W. Felts, and J. Nelson. "A Man-Machine Approach Toward Solving the Generalized Truck-Dispatching Problem." Transportation Science, 6, 21 (May, 1972), p. 149.
- (17) Kuehnle, H., and R. Mappes. "The State-of-the-Art in Computer Scheduling of Buses." School Bus Fleet (1969), p. 24.
- (18) Lin, S. "Computer Solutions of the Traveling Salesman Problem." Bell Systems Technical Journal, 44 (1965), pp. 2245-2269.
- (19) Little, J. D. C., K. G. Murty, D. W. Sweeney, and C. Karel. "An Algorithm for the Traveling Salesman Problem." Operations Research, 11 (1963), pp. 972-989.
- (20) Robbins, J. "The Clarke and Wright-Lin Routing Algorithm: Analysis of an Added Restriction." (Unpublished working paper, Industrial Engineering and Management Department, Oklahoma State University, January, 1976.)
- (21) Robbins, J., J. Shamblyn, W. Turner, and D. Byrd. "Development of and Computational Experience with a Combination Tour Construction—Tour Improvement Algorithm for Vehicle Routing Problems." (Presented at the ORSA/TIMS meeting, Las Vegas, Nevada, November, 1975.)
- (22) Svestka, J., and V. E. Huckfeldt. "Computational Experience with an M-Salesman Traveling Salesman Algorithm." Management Science, 19, 7 (March, 1973), pp. 790-799.
- (23) Tillman, F., and H. Cochran. "A Heuristic Approach for Solving the Delivery Problem." J. Ind. Engineering, 19 (1968), p. 354.
- (24) Turner, W. C., L. Foulds, and P. M. Ghare. "Transportation Routing Problem—A Survey." AIIE Transactions, 6, 4 (1974), pp. 288-301.



## APPENDIX A

### USER INSTRUCTIONS

The first step in reading in the required data is to number each node starting at one and ending with N, where N is the total number of nodes excluding the depot. The data must be read in the following order:

- (1) Number of stops and vehicle capacity;
- (2) Demand at each node; and
- (3) Distance matrix.

(1) Number of stops and vehicle capacity.

The number of stops (NSTOP) and vehicle capacity (NCAP) are the first variables read, respectively. Both variables are INTEGER and must be read according to the format 2I10.

(2) Demand at each node.

The demand at each node is read into the INTEGER array NUSTUD. Sixteen demands are put on each data card as the format is 16I5. Demands must be read in from node one to N. Continuation cards must be used for problems containing more than sixteen nodes.

(3) Distance matrix.

The symmetric distance matrix is read into a dummy real variable called DUMY. Sixteen distances are put on each data card as the format is 16F5.2. Continuation cards must be used for problems containing

more than sixteen nodes. Only the top half of a distance matrix is read in. That is, for stop  $i$  the distances  $i$  to  $i + 1$ ,  $i$  to  $i + 2$ ,  $i$  to  $i + 3$ , ...,  $i$  to  $N$  are used. Thus, for programming purposes, the two matrices below are equivalent.

$$\begin{array}{c}
 \begin{array}{cccc}
 & 1 & 2 & 3 & 4 \\
 1 & \left[ \begin{array}{cccc}
 0 & 3.5 & 5 & 1 \\
 3.5 & 0 & 2 & 8 \\
 5 & 2 & 0 & 4 \\
 1 & 8 & 4 & 0
 \end{array} \right] \\
 2 \\
 3 \\
 4
 \end{array}
 &
 \begin{array}{c}
 \begin{array}{cccc}
 & 1 & 2 & 3 & 4 \\
 1 & \left[ \begin{array}{ccc}
 & 3.5 & 5 & 1 \\
 & & 2 & 8 \\
 & & & 4 \\
 & & & & 
 \end{array} \right] \\
 2 \\
 3 \\
 4
 \end{array}
 \end{array}
 \end{array}$$

The use of half the distance matrix saves keypunch and computer read in time. Note the distance from any node to itself is zero. This notation is used for an infeasible route. If any links are deemed infeasible, the zero distance should be used.

#### Example

The data cards for an eight node routing problem are to be prepared. Vehicle capacity is twenty units while the node demands are shown in Table IX. The distance matrix is given in Table X. The nodes are numbered one through eight with the depot numbered nine. Table XI shows the data cards that must be prepared and read into the computer.

Card one contains the number of stops in column ten (NSTOP=8) and the vehicle capacity in columns nineteen and twenty (NCAP=20). Card two contains the demand at each node. Columns four and five contain the demand at node one (NUSTUD(1)=10), column ten has the demand at

TABLE IX  
NODE DEMANDS

Node	Demand	Node	Demand
1	10	5	4
2	3	6	15
3	8	7	8
4	7	8	5

TABLE X  
NODE DISTANCES

	1	2	3	4	5	6	7	8	9
1	0	4.50	2.0	9.0	15.75	1.25	0	8.0	3.0
2	4.50	0	6.25	7.75	11.50	3.0	3.0	1.0	4.0
3	2.0	6.25	0	0	2.50	9.50	14.0	5.0	1.0
4	9.0	7.75	0	0	1.0	11.25	0	5.50	5.0
5	15.75	11.50	2.50	1.0	0	9.50	8.0	3.0	4.0
6	1.25	3.0	9.50	11.25	9.50	0	2.50	0	10.0
7	0	3.0	14.0	0	8.50	2.50	0	1.50	4.50
8	8.0	1.0	5.0	5.50	3.0	0	1.50	0	12.0
9	3.0	4.0	1.0	5.0	4.0	10.0	4.50	12.0	0

TABLE XI  
DATA CARDS

Column	11111111111222222222233333333334	8
Card	1234567890123456789012345678901234567890	... 0
1	8	20
2	10	3 8 7 4 15 8 5
3	4.5	2. 9. 15.75 1.25 . 8. 3.
4	6.25	7.75 11.5 3. 3. 1. 4.
5	.	2.5 9.5 14. 5. 1.
6	1.	11.25 . 5.5 5.
7	9.5	8. 3. 4.
8	2.5	. 10.
9	1.5	4.5
10	12.	

node two (NUSTUD(2)=3), etc. If the problem had contained more than sixteen nodes, the demands for nodes 17, 18, ..., N would be placed, in the same format, on continuation cards.

The distances from Table X are recorded on cards three through ten. Note that only the top half of the matrix is used. For example, card three contains the distances from node one to nodes two through nine where node nine is the depot. Card four contains the distances from node two to nodes three through nine. For problems larger than sixteen nodes, continuation cards must be used for all nodes past the sixteenth node.

The storage requirements for usage of the program must be watched very carefully. Naturally, the larger the number of nodes, the larger the storage requirements. If a problem should become too large for a computer in terms of the storage needed, the program could be run in two parts (see Chapter VII). Should the problem still be too large, a computer with more storage capability should be used. Other alternatives are explored in Chapter VII.

DIMENSION statements are used by the program to set up arrays. There are nine subroutines used and thus nine sets of DIMENSION statements. Each set contains four cards of which three can be used for any problem up to three hundred and fifty nodes and one hundred routes. The remaining card in each set must be changed each time the number of nodes changes. This card, for fifty nodes, appears below:

```
DIMENSION DIST(51,51),ISAVE(51,51),IX(2601),DUMY(51)
```

Note the arguments of the arrays are fifty-one, not fifty. This is because the depot must be included. Should a problem with one hundred

nodes be solved after the fifty node problem, the DIMENSION card above must be replaced with the following:

```
DIMENSION DIST(101,101),ISAVE(101,101),IX(10201),DUMY(101)
```

For problems of over three hundred and fifty nodes or one hundred routes, all DIMENSION statements must be replaced.

APPENDIX B

COMPUTER PROGRAM

```

C *****
C *****
C THIS PROGRAM CREATES ROUTES BY THE CLARK-WRIGHT PROCEDURE AND
C IMPROVES THE ROUTES WITH A 2-OPT WITHIN ROUTE SWAPING PROCEDURE.
C THIS PROGRAM WILL HANDLE, AS DIMENSIONED, 400 STOPS AND 100 ROUTES.
C THE ONLY ITEM THAT MUST BE CHANGED IS THE DIMENSION STATEMENT
C CONTAINING 'DIST', 'ISAVE', 'IX', AND 'DUMY'. THESE VARIABLES MUST
C BE SUBSCRIPTED ACCORDING TO THE NUMBER OF STOPS PLUS HOMEBASE IN THE
C PROBLEM. 'IX' IS EQUAL TO : (NUMBER OF STOPS + HOMEBASE)**2.
C *****
C *****
C
C MAN: TELLS THE COMPUTER IF THE INPUTS ARE FOR THE INTERACTION PHASE
C NSTOP: NUMBER OF STOPS IN SYSTEM
C NCAP: CAPACITY OF VEHICLE
C NSTUD(I): NUMBER OF STUDENTS AT STOP I
C NUSTU(J): NUMBER OF STUDENTS ON THE J'TH ROUTE
C DIST(I,J): DISTANCE BETWEEN STOPS I AND J
C SET: ARRAY THAT KEEPS TRACK OF THE STOPS AND WHICH ROUTE THEY ARE ON.
C THE FIRST LOCATION OF THE ARRAY TELLS THE STOP NUMBER
C AND THE SECOND ARG TELLS WHICH ROW OF SET THE FOLLOWING
C STOP SITS IN.
C IROW(I): POINTER; 0= HAVE NOT TRAVELLED FROM I TO SOME J.
C >0= HAVE TRAVELLED FROM I TO SOME J
C IROW(I) ALSO TELLS WHICH ROW OF SET STOP I SITS IN.
C JCCL(J): POINTER; =0 MEANS HAVE NOT TRAVELLED FROM SOME I TO J.
C >0 MEANS HAVE PREVIOUSLY TRAVELLED FROM SOME I TO J
C JCCL(J) ALSO TELLS WHICH ROW OF SET THE STOP AHEAD OF J
C SITS IN ONCE J HAS BEEN TRAVELLED TO.
C NXTROW: THE NEXT ROW OF SET TO BE FILLED.
C DISROU(I): DISTANCE OF ROUTE I
C NXTRT: NEXT ROUTE TO PUT STUDENTS ON. (NXTRT-1) ROUTES EXIST ALREADY
C NNSTPS(J) # OF STOPS ON THE J'TH ROUTE
C
C INTEGER*2 ISAVE,IROUTE,IROW,JCCL,SET,NBEG,NEND,NUSTU,NSTUD,IX, M35
C 1DIST,NNSTPS
C DIMENSION DIST(101,101),ISAVE(101,101),IX(10201),DUMY(101),
C ISET(400,2)
C DIMENSION NSTUD(400),NUSTU(100),NEND(100),NBEG(100),NNSTPS(100),
C 1IROW(400),JCCL(400),IROUTE(400),DISROU(100)
C COMMON DISROU,DUMY,NXTROW,NXTRT,NCAP,IX,NNSTPS,IROUTE,NUSTU,NSTUD,M41
C 1NBEG,NEND,JCCL,IROW,SET,DIST,ISAVE M42

```

```

C
C   READ INPUT DATA
C
  READ(5,10)NSTOP,NCAP,MAN
10  FORMAT(3I10)
  WRITE(6,9)NSTOP,NCAP
  9  FORMAT('1',T10,'NUMBER OF STOPS IN SYSTEM: ',I3,/,T10,
1   'CAPACITY OF VEHICLE: ',I3,/)
  READ(5,15)(NSTUD(I),I=1,NSTOP)
15  FORMAT(16I5)
  NENTRY=NSTOP+1
  DO 22 I=1,NSTOP
  IROW(I)=0
  JCCL(I)=0
  DIST(I,I)=0
  IL=I+1
  READ(5,16)(DUMY(J),J=I1,NENTRY)
16  FORMAT(16F5.1)
  DO 21 J=I1,NENTRY
  DIST(I,J)=DUMY(J)*100.+5
21  DIST(J,I)=DIST(I,J)
22  CONTINUE
  DIST(NENTRY,NENTRY)=0
  IF(MAN.EQ.0)GO TO 25
  CALL INPUT(NENTRY)
  DO 24 I=1,NXTRT
  IF(NUSTU(I).LE.NCAP)GO TO 24
  WRITE(6,23)I,NUSTU(I)
23  FORMAT(/////T10,'***** ROUTE ',I3,' IS OVERLOADED WITH ',I3,' STU
  IDENTS *****')
  GO TO 2
24  CONTINUE
  GO TO 72

C
C   COMPUTE SAVINGS
C
25  CONTINUE
  INC=0

C
C   CALCULATE 'INC', THE NUMBER OF FEASIBLE ARCS
C
  DO 30 I=1,NSTOP
  DO 30 J=I,NSTOP
  ISAVE(I,J)=-9999
  IF(DIST(I,J).EQ.0 )GO TO 29
  INC=INC+2
  ISAVE(I,J)=DIST(I,NENTRY)+DIST(NENTRY,J)-DIST(I,J)
29  CONTINUE
  ISAVE(J,I)=ISAVE(I,J)
30  CONTINUE
  DO 31 I=1,NENTRY
  ISAVE(I,NENTRY)=-9999
  ISAVE(NENTRY,I)=ISAVE(I,NENTRY)
31  CONTINUE

C
C   ORDER SAVINGS IN DESCENDING ORDER
C
  N=NENTRY*NENTRY
  CALL SORT(N,ISAVE,IX)
C

```

M47  
M48



```

C   BUILD ROUTES
C
  NXTRT=1
  NXTROW=1
  DO 60 K=1, INC
  IXK=IX(K)
  I=IXK-(IXK/NENTRY)*NENTRY
C
C   CALCULATING I AND J   TRAVEL FROM HOME TO I OR J IS PROHIBITED
C
  IF(I.EQ.0)GO TO 60
  IF(IROW(I).GT.0)GO TO 60
  J=(IXK-1)/NENTRY+1
  IF(J.EQ.NENTRY)GO TO 60
C
C   CHECK POINTERS TO SEE IF I OR J HAS BEEN INCLUDED IN PREVIOUS ROUTE
C
  IF(JCOL(J).GT.0)GO TO 60
  IF(JCOL(I).EQ.0)GO TO 45
  IF(IROW(J).EQ.0)GO TO 36
  CALL CCMBNE(I,J)
C
C   MAY BE ABLE TO JOIN TWO ROUTES
C
  GO TO 60
36 CALL ADD(I,J,0)
C
C   ADD J ON END OF A ROUTE
C
  GO TO 60
45 IF (IROW(J).EQ.0)GO TO 50
  CALL ADD(J,I,1)
C
C   ADD I ON FRONT OF A ROUTE
C
  GO TO 60
50 CALL NWROUT(I,J)
C
C   START NEW ROUTE
C
60 CONTINUE
C
C   FINISH BUILDING ROUTES   NOW PRINT THEM OUT
C
  NXTRT=NXTRT-1
72 CONTINUE
  IF(NXTROW-1.NE.NSTCP)WRITE(6,78)
78 FORMAT(////'ALL STOPS ARE NOT INCLUDED IN THE ROUTES'////)
C
C   PLACE HOME BASE IN ARRAY CALLED SET AT BEGINNING AND END OF ROUTE.
C   SET UP POINTERS.
C
  TDIS=0.
  DO 75 KL=1,NXTRT
  M=NEND(KL)
  J=SET(M,1)
  IROW(J)=M
  SET(M,2)=NXTROW
  SET(NXTROW,1)=NENTRY
  NEND(KL)=NXTROW

```

```
NXTROW=NXTROW+1
M=NBEG(KL)
SET(NXTROW,1)=NENTRY
SET(NXTROW,2)=M
J=SET(M,1)
JCOL(J)=NXTROW
IROW(NENTRY)=NXTROW
NBEG(KL)=NXTROW
NXTROW=NXTROW+1
NNSTPS(KL)=NNSTPS(KL)+2
CALL PRTOUT(KL,NENTRY)
TDIS=TDIS+DISRCU(KL)
WRITE(6,74)
74 FORMAT(///)
75 CONTINUE
WRITE(6,1)TDIS
1 FORMAT(10X,'TOTAL DISTANCE OF ROUTES = ',F10.2)
C
C
C CALL WITHIN ROUTE SWAPPING SUBROUTINE
CALL WRT2OP(NENTRY)
2 CONTINUE
RETURN
END
```

```

SUBROUTINE ADD(L,J,NKEY)
  INTEGER*2 ISAVE,IROUTE,IROW,JCOL,SET,NBEG,NEND,NUSTU,NSTUD,IX,
  IDIST,NNSTPS
  DIMENSION DIST(101,101),ISAVE(101,101),IX(10201),DUMY(101),
  ISET(400,2)
  DIMENSION NSTUD(400),NUSTU(100),NEND(100),NBEG(100),NNSTPS(100),
  IROW(400),JCOL(400),IROUTE(400),DISROU(100)
  COMMON DISROU,DUMY,NXTROW,NXTRT,NCAP,IX,NNSTPS,IROUTE,NUSTU,NSTUD,AB
  INBEG,NEND,JCCL,IROW,SET,DIST,ISAVE
  A9
C
C   ADD J TO L'S ROUTE   NKEY SAYS EITHER BACK OR FRONT OF ROUTE
C
  K=IROUTE(L)
  IF(NUSTU(K)+NSTUD(J).GT.NCAP)GO TO 20
  IROUTE(J)=K
  NUSTU(K)=NSTUD(J)+NUSTU(K)
  NNSTPS(K)=NNSTPS(K)+1
  IF(NKEY.EQ.1)GO TO 10
C
C   ADD ON END OF ROUTE
C
  M=NEND(K)
  SET(M,2)=NXTROW
  NEND(K)=NXTRCW
  IROW(L)=M
  JCOL(J)=M
  WRITE(6,11)J,K
  11 FORMAT(/T10,'ADD STOP ',I3,' ON END   OF ROUTE ',I3)
  GO TO 15
C
C   ADD ON FRGNT OF ROUTE
C
  10 SET(NXTROW,2)=NBEG(K)
  NBEG(K)=NXTROW
  IROW(J)=NXTRCW
  JCOL(L)=NXTRCW
  WRITE(6,12)J,K
  12 FORMAT(/T10,'ADD STOP ',I3,' ON FRONT OF ROUTE ',I3)
  15 SET(NXTROW,1)=J
  NXTROW=NXTROW+1
  20 CONTINUE
  RETURN
  END

```

```

SUBROUTINE NWROUT(I,J)
C
C   THIS SUBROUTINE CREATES A NEW ROUTE
C
C   THE NEW ROUTE, NUMBERED 'NXTRT' CONTAINS, INITIALLY, TWO STOPS, I-J.
C
  INTEGER*2 ISAVE,IRCUTE,IROW,JCOL,SET,NBEG,NEND,NUSTU,NSTUD,IX,
  IDIST,NNSTPS
  DIMENSION DIST(101,101),ISAVE(101,101),IX(10201),DUMY(101),
  ISET(400,2)
  DIMENSION NSTUD(400),NUSTU(100),NEND(100),NBEG(100),NNSTPS(100),
  IROW(400),JCOL(400),IROUTE(400),DISROU(100)
  COMMON DISROU,DUMY,NXTROW,NXTRT,NCAP,IX,NNSTPS,IROUTE,NUSTU,NSTUD,N13
  1NBEG,NEND,JCOL,IROW,SET,DIST,ISAVE                                N14
C
C   UPDATING POINTERS
C
  SET(NXTROW,1)=I
  SET(NXTROW,2)=NXTROW+1
  NBEG(NXTRT)=NXTROW
  IROW(I)=NXTROW
  JCOL(J)=NXTROW
  NXTROW=NXTROW+1
  SET(NXTROW,1)=J
  NEND(NXTRT)=NXTROW
  NXTROW=NXTROW+1
  IROUTE(I)=NXTRT
  IROUTE(J)=NXTRT
  NUSTU(NXTRT)=NSTUD(I)+NSTUD(J)
  NNSTPS(NXTRT)=2
  WRITE(6,10)I,J,NXTRT
10 FORMAT(/,T10,'STOPS ',I4,' AND ',I4,' FORM NEW ROUTE ',I3)
  NXTROW=NXTROW+1
  RETURN
  END

```

```

SUBROUTINE COMBNE(I,J)
  INTEGER*2 ISAVE,IRCUTE,IROW,JCOL,SET,NBEG,NEND,NUSTU,NSTUD,IX,
  IDIST,NNSTPS
  DIMENSION DIST(101,101),ISAVE(101,101),IX(10201),DUMY(101),
  ISET(400,2)
  DIMENSION NSTUD(400),NUSTU(100),NEND(100),NBEG(100),NNSTPS(100),
  IROW(400),JCOL(400),IRUTE(400),DISROU(100)
  COMMON DISROU,DUMY,NXTROW,NXTRT,NCAP,IX,NNSTPS,IRUTE,NUSTU,NSTUD,C8
  INBEG,NEND,JCOL,IROW,SET,DIST,ISAVE C9
C
C THIS SUBROUTINE LINKS TWO ROUTES TOGETHER AND UPDATES THE OTHER
C ROUTE POINTERS
C
C   IF(IRUTE(I).EQ.IRUTE(J))GO TO 9
C
C   MUST BE SURE NOT TO FORM A LOOP
C
C   KK=IRUTE(I)
C   LL=IRUTE(J)
C
C   CHECK TO SEE IF JOINING TWO ROUTES WILL EXCEED CAPACITY
C
C   IF(NUSTU(KK)+NUSTU(LL).GT.NCAP)GO TO 9
C
C   NXTRT IS TEMPORARILY USED TO DENOTE THE NUMBER OF EXISTING ROUTES
C   AFTER JOINING TWO ROUTES
C
C   NXTRT=NXTRT-2
C
C   UPDATING POINTERS
C
C   M=NEND(KK)
C   L=NBEG(LL)
C   SET(M,2)=L
C   IROW(I)=M
C   JCOL(J)=M
C   NEND(KK)=NEND(LL)
C   NUSTU(KK)=NUSTU(KK)+NUSTU(LL)
C   NNSTPS(KK)=NNSTPS(KK)+NNSTPS(LL)
C   NJ=NNSTPS(LL)
C   CALL FIXUP(KK,LL,NJ,L)
C
C   KK: ROUTE NUMBER STOP I IS IN
C   LL: ROUTE NUMBER STOP J IS IN
C   NJ: NUMBER OF STOPS IN J'S ROUTE
C   L: ROW OF ARRAY 'SET' THAT THE FIRST STOP IN J'S ROUTE SITS IN
C
C   IF(LL.GT.NXTRT)GO TO 8
C
C   CHECKING TO SEE IF WE HAVE PUT THE LAST ROUTE ON THE END OF ANCTHER
C   ROUTE, IF SO NO NEED TO CHANGE POINTERS.
C
C   DO 5 JJ=LL,NXTRT
C     JK=JJ+1
C     N=NNSTPS(JK)
C     L=NBEG(JK)
C     CALL FIXUP(JJ,JK,N,L)
C     NEND(JJ)=NEND(JK)
C     NBEG(JJ)=NBEG(JK)

```

```
    NUSTU(JJ)=NUSTU(JK)
    NNSTPS(JJ)=NNSTPS(JK)
5  CONTINUE
8  CONTINUE
    WRITE(6,10)KK,LL,KK
10 FORMAT(/T10,'ROUTES ',I3,' AND ',I3,' COMBINE TO FORM NEW ROUTE',
113, //T10,'UPDATE ROUTE NUMBERS')
    NXTRT=NXTRT+1
9  CONTINUE
    RETURN
    END
```

```

SUBROUTINE FIXUP(KK,LL,N,L)
C
C KK: ROUTE # STOP I SI IN.
C LL: ROUTE # STOP J SI IN.
C N: NUMBER OF STOPS IN J'S ROUTE.
C L: ROW OF ARRAY 'SET' THAT THE FIRST STOP IN J'S ROUTE SITS IN.
C
  INTEGER*2 ISAVE,IROUTE,IROW,JCOL,SET,NBEG,NEND,NUSTU,NSTUD,IX,
  IDIST,NNSTPS
  DIMENSION DIST(101,101),ISAVE(101,101),IX(10201),DUMY(101),
  ISET(400,2)
  DIMENSION NSTUD(400),NUSTU(100),NEND(100),NBEG(100),NNSTPS(100),
  IROW(400),JCOL(400),IROUTE(400),DISROU(100)
  COMMON DISROU,DUMY,NXTROW,NXTRT,NCAP,IX,NNSTPS,IROUTE,NUSTU,NSTUD,
  1NBEG,NEND,JCOL,IROW,SET,DIST,ISAVE
C
C THIS SUBROUTINE REORDERS THE POINTERS AFTER TWO ROUTES ARE
C COMBINED INTO ONE
C
  DO 5 NT=1,N
    MM=SET(L,1)
    IROUTE(MM)=KK
    IF(NT.EQ.N)GO TO 5
    L=SET(L,2)
  5 CONTINUE
  RETURN
  END

```

```

SUBROUTINE SORT(N, X, NDX)
C
C N: TOTAL NUMBER OF LOCATIONS TO BE ORDERED.(NENTRY**2)
C X: ARRAY TO BE ORDERED FROM HIGHEST TO LOWEST. ARRAY IS THE
C SAVINGS ARRAY
C NDX: ARRAY CONTAINING THE SAVINGS IN ORDER. NDX IS SINGLE DIMENSIONED
C WHILE ISAVE IS TWO DIMENSION.
C
C INTEGER*2 X,NDX S9
C DIMENSION X(1),NDX(1) S10
C
C FAST SORT ROUTINE
C RETURNS SORTED ARRAY OF POINTERS 'NOX' DOES NOT EFFECT 'X'
C
DO 10 I =1,N
10 NDX(I) = I
M = N
20 M = M/2
IF(M .EQ. 0) GO TO 80
30 K = N-M
J = 1
40 I = J
50 L = I+M
NI = NDX(I)
NL = NDX(L)
IF( X(NI)-X(NL) .GE. 0) GO TO 70
60 NDX(I) = NL
NDX(L) = NI
I = I-M
IF(I-1 .GE. 0) GO TO 50
70 J = J+1
IF (J-K) 40,40,20
80 CONTINUE
RETURN
END

```



```

SUBROUTINE WRT2OP(NENTRY)
  INTEGER*2 ISAVE, IROUTE, IROW, JCOL, SET, NBEG, NEND, NUSTU, NSTUD, IX,
  1DIST, NNSTPS
  DIMENSION DIST(101,101), ISAVE(101,101), IX(10201), DUMY(101),
  1SET(400,2)
  DIMENSION NSTUD(400), NUSTU(100), NEND(100), NBEG(100), NNSTPS(100),
  1IRDW(400), JCOL(400), IROUTE(400), DISROU(100)
  COMMON DISROU, DUMY, NXTROW, NXTRT, NCAP, IX, NNSTPS, IROUTE, NUSTU, NSTUD,
  1NBEG, NEND, JCOL, IROW, SET, DIST, ISAVE

```

C  
C  
C  
C  
C  
C  
C  
C  
C  
C  
C

THIS SECTION DEALS WITH IMPROVING AN EXISTING ROUTE BY USING THE  
TWO OPT PROCEDURE, WITHIN ROUTES.

THIS SUBROUTINE CHECKS FOR SWAPING OF STOPS(I,J) AND (K,L) SO WE GET  
NEW STOP ORDER (I,K) AND (J,L).

```

  WRITE(6,6)
  6 FORMAT('1')

```

C  
C  
C  
C  
C  
C  
C  
C  
C  
C  
C  
C  
C

NXTRT IS NOW THE NUMBER OF ROUTES IN THE SYSTEM  
NNSTPS: NUMBER OF STOPS IN ROUTE WITH HOME COUNTED TWICE  
NSTPS IS THE NUMBER OF LINKS IN THE KL'TH ROUTE.  
NFLNKS: THE NUMBER OF FEASIBLE LINKS IN THE KL'TH ROUTE  
NUMBER OF FEASIBLE LINKS TO COMPARE (I,J) TO EQUALS NUMBER OF STOPS  
WITH HOME COUNTED TWICE, MINUS THREE.

TDIS: KEEPS TRACK OF THE TOTAL DISTANCE TRAVELLED BY ALL ROUTES.

```

  TDIS=0.
  DO 1000 KL=1, NXTRT
    COST=0.
    N=NNSTPS (KL)
    NSTPS=N-1
    NFLNKS=N-3
    IF(NFLNKS.LE.0)GO TO 1000
104 CONTINUE
    NP=NBEG(KL)
    DO 107 J=1, NFLNKS
      NI=SET(NP,1)
      NP=SET(NP,2)
      NJ=SET(NP,1)
      L1=J+2
      NPP=SET(NP,2)
      DO 106 I=L1, NSTPS
        NK=SET(NPP,1)
        NPP=SET(NPP,2)
        NL=SET(NPP,1)
        IF(NI.EQ.NL)GO TO 107
        IF(DIST(NI,NK).EQ.0.OR.DIST(NJ,NL).EQ.0)GO TO 106

```

C  
C  
C  
C

CHECKING THE FEASIBILITY OF SWAPING THE LINKS OF (I,J) AND (K,L) TO  
(I,K) AND (J,L)

```

  NCOST=DIST(NI, NJ)+DIST(NK, NL)-DIST(NI, NK)-DIST(NJ, NL)

```

```
IF(NCOST.LE.C)GO TO 106
COST=COST+NCOST
K=I+1
CALL REOSTP(NI,NJ,NK,NL,KL,NENTRY)
GO TO 104
106 CONTINUE
107 CONTINUE
COST=COST/100.
CALL PRTOU(KL,NENTRY)
TDIS=TDIS+DISROU(KL)
C
C COST CONTAINS THE SAVINGS REALIZED BY THE 2-OPT WITHIN ROUTE
C SWAPING FOR THE KL*TH ROUTE.
C
WRITE(6,7)COST
7 FORMAT('+',T12,'SAV= ',F6.1,///)
1000 CONTINUE
WRITE(6,1)TDIS
1 FORMAT(10X,'TOTAL DISTANCE OF RCUTES = ',F10.2)
RETURN
END
```

```

SUBROUTINE REOSTP(I,J,K,L,KL,NENTRY)

```

```

C
C
C
C
C
C
C
C
C

```

```

THIS SUBROUTINE REORDERS THE STOPS TRAVELLED ON THE KL*TH ROUTE.
ONCE I,J,K, AND L ARE KNOWN THE REORDERING TAKES PLACE.
LET I=1, J=8, K=4, AND L=3. THE ROUTE H-7-9-1-8-2-10-5-4-3-H
BECOMES H-7-9-1-4-5-10-2-8-3-H WHERE THE SEQUENCE
8-2-10-5-4 IS REVERSED.

```

```

INTEGER*2 ISAVE,IROUTE,IROW,JCCL,SET,NBEG,NEND,NUSTU,NSTUD,IX,
1DIST,NNSTPS
DIMENSION DIST(101,101),ISAVE(101,101),IX(10201),DUMY(101),
1SET(400,2)
DIMENSION NSTUD(400),NUSTU(100),NEND(100),NBEG(100),NNSTPS(100),
1IROW(400),JCCL(400),IROUTE(400),DISROU(100)
COMMON DISROU,DUMY,NXTROW,NXTRT,NCAP,IX,NNSTPS,IROUTE,NUSTU,NSTUD,
1NBEG,NEND,JCCL,IROW,SET,DIST,ISAVE

```

```

C
C
C

```

```

REARRANGE POINTERS

```

```

IF(I.NE.NENTRY)GO TO 2
M=NBEG(KL)
GO TO 3
2 M=IROW(I)
3 N=IROW(K)
5 SET(M,2)=N
II=SET(N,1)
JJ=JCCL(II)
JCOL(II)=M
IF(II.EQ.J)GO TO 6
M=N
N=JJ
GO TO 5
6 IF(L.EQ.NENTRY)GO TO 8
SET(N,2)=IROW(L)
GO TO 9
8 SET(N,2)=NEND(KL)
9 JCCL(L)=IROW(J)
RETURN
END

```

```

SUBROUTINE PRTOU(KL,NENTRY)
  INTEGER*2 ISAVE, IROUTE, IROW, JCOL, SET, NBEG, NEND, NUSTU, NSTUD, IX,
  IDIST, NNSTPS
  DIMENSION DIST(101,101), ISAVE(101,101), IX(10201), DUMY(101),
  ISET(400,2)
  DIMENSION NSTUD(400), NUSTU(100), NEND(100), NBEG(100), NNSTPS(100),
  IROW(400), JCOL(400), IROUTE(400), DISROU(100)
  COMMON DISROU, DUMY, NXTROW, NXTRT, NCAP, IX, NNSTPS, IROUTE, NUSTU, NSTUD,
  INBEG, NEND, JCOL, IROW, SET, DIST, ISAVE
C
C PRINTS OUT THE STOPS, STUDENTS, AND DISTANCE TRAVELLED ON THE KL*TH
C ROUTE.
C
  WRITE(6,70)
70 FORMAT(T 8,'ROUTE',T16,'STUDENTS',T31,'DISTANCE',/)
  M=NNSTPS(KL)
  DISROU(KL)=0.
  N=M-1
  NP=NBEG(KL)
  J=SET(NP,1)
C
C PLACE STOPS OF EACH ROUTE IN ARRAY SO THEY CAN BE WRITTEN OUT
C
  DO 73 KH=1,N
  IX(KH)=J
  NP=SET(NP,2)
71 I=J
  J=SET(NP,1)
  DISROU(KL)=DISROU(KL)+DIST(I,J)
73 CONTINUE
  DISROU(KL)=DISROU(KL)/100.
  IX(N+1)=NENTRY
C
C REDUCE DISTANCE BY 100 SO CAN USE INTEGER*2
C
  WRITE(6,6)KL,NUSTU(KL),DISROU(KL),(IX(I),I=1,M)
6 FORMAT(2I10,T30,F6.1,/,1X,32I4)
  RETURN
  END

```

```

SUBROUTINE INPUT(NENTRY)
  INTEGER*2 ISAVE, IROUTE, IROW, JCOL, SET, NBEG, NEND, NUSTU, NSTUD, IX,
  1DIST, NNSTPS
  DIMENSION DIST(101,101), ISAVE(101,101), IX(10201), DUMY(101),
  1SET(400,2)
  DIMENSION NSTUD(400), NUSTU(100), NEND(100), NBEG(100), NNSTPS(100),
  1IROW(400), JCOL(400), IROUTE(400), DISROU(100)
  COMMON DISROU, DUMY, NXTROW, NXTRT, NCAP, IX, NNSTPS, IROUTE, NUSTU, NSTUD,
  1NBEG, NEND, JCOL, IROW, SET, DIST, ISAVE
  WRITE(6,10)
10 FORMAT(///,T30,'***** INTERACTION PHASE INPUT *****',///)
  READ(5,1)NXTRT
  1 FORMAT(I10)
  NXTROW=1
  DO 9 I=1,NXTRT
    NUSTU(I)=0
    READ(5,2)N, (IX(J),J=1,N)
  2 FORMAT(1X,I3,19I4)
    NNSTPS(I)=N
    DO 9 J=1,N
      M=IX(J)
      SET(NXTROW,1)=M
      IROW(M)=NXTROW
      IF(J.NE.1)GO TO 3
      NBEG(I)=NXTROW
      GO TO 4
  3 JCOL(M)=NXTROW-1
  4 NUSTU(I)=NUSTU(I)+NSTUD(M)
      M=NXTROW
      NXTROW=NXTROW+1
      IF(J.EQ.N)GO TO 8
      SET(M,2)=NXTROW
      GO TO 9
  8 NEND(I)=M
  9 CONTINUE
  RETURN
  END

```

APPENDIX C

COORDINATES AND DEMANDS FOR SIX RANDOMLY  
GENERATED PROBLEMS

TABLE XII

PROBLEM 1

Site	x	y	Demand
1	68.	52.	10
2	50.	71.	7
3	72.	40.	5
4	38.	15.	9
5	98.	87.	8
6	63.	55.	3
7	68.	11.	4
8	80.	62.	1
9	99.	99.	7
10	58.	56.	7

Depot coordinates (50,50).

Capacity of vehicle: 50 units.

TABLE XIII

## PROBLEM 2

Site	x	y	Demand
1	53.	6.	9
2	77.	5.	11
3	83.	52.	4
4	64.	30.	3
5	65.	98.	6
6	86.	26.	10
7	38.	12.	11
8	37.	28.	4
9	44.	99.	4
10	27.	78.	5
11	95.	31.	11
12	51.	17.	10
13	45.	52.	7
14	74.	91.	2
15	83.	30.	5
16	45.	31.	7
17	14.	77.	2
18	59.	66.	6
19	3.	35.	4
20	20.	83.	8
21	70.	24.	2
22	66.	82.	3
23	13.	99.	7
24	42.	70.	5
25	35.	44.	2
26	88.	47.	8
27	11.	91.	11
28	43.	70.	6
29	1.	78.	6
30	11.	39.	7

Depot coordinates (50,50).

Capacity of vehicle: 50 units.

TABLE XIV

## PROBLEM 3

Site	x	y	Demand	Site	x	y	Demand
1	29.	84.	7	26	80.	6.	6
2	23.	89.	11	27	47.	24.	9
3	8.	64.	4	28	54.	6.	11
4	95.	21.	6	29	78.	66.	11
5	38.	29.	8	30	8.	18.	3
6	77.	48.	9	31	3.	64.	11
7	36.	31.	10	32	74.	79.	2
8	3.	66.	9	33	9.	77.	11
9	46.	90.	7	34	37.	65.	11
10	35.	20.	4	35	39.	92.	8
11	29.	35.	9	36	0.	29.	9
12	9.	61.	11	37	65.	4.	7
13	64.	13.	10	38	29.	78.	11
14	63.	39.	8	39	97.	77.	5
15	77.	25.	3	40	4.	56.	11
16	79.	5.	7	41	29.	27.	3
17	64.	96.	8	42	30.	3.	7
18	95.	90.	4	43	83.	48.	5
19	1.	93.	8	44	44.	25.	5
20	74.	82.	11	45	39.	17.	3
21	1.	4.	4	46	46.	97.	9
22	15.	13.	2	47	38.	39.	5
23	46.	91.	3	48	18.	34.	3
24	71.	48.	8	49	85.	62.	9
25	54.	34.	2	50	41.	70.	8

Depot coordinates (50,50).

Capacity of vehicle: 50 units.



TABLE XV

## PROBLEM 4

Site	x	y	Demand	Site	x	y	Demand
1	64.	96.	3	36	67.	99.	2
2	80.	39.	2	37	48.	83.	5
3	69.	23.	7	38	75.	81.	4
4	72.	42.	10	39	8.	19.	8
5	48.	67.	5	40	20.	18.	2
6	58.	43.	9	41	54.	38.	3
7	81.	34.	2	42	63.	36.	5
8	79.	17.	5	43	44.	33.	2
9	30.	23.	6	44	52.	18.	6
10	42.	67.	3	45	12.	13.	5
11	7.	76.	2	46	25.	5.	2
12	29.	51.	3	47	58.	85.	5
13	78.	92.	11	48	5.	67.	2
14	64.	8.	2	49	90.	9.	11
15	95.	57.	3	50	41.	76.	8
16	57.	91.	5	51	25.	76.	2
17	40.	35.	7	52	37.	64.	3
18	68.	40.	4	53	56.	63.	2
19	92.	34.	3	54	10.	55.	10
20	62.	1.	7	55	98.	7.	9
21	28.	43.	2	56	16.	74.	8
22	76.	73.	8	57	89.	60.	7
23	67.	88.	6	58	48.	82.	9
24	93.	54.	7	59	81.	76.	6
25	6.	8.	7	60	29.	60.	9
26	87.	18.	6	61	17.	22.	3
27	30.	9.	6	62	5.	45.	10
28	77.	13.	5	63	79.	70.	3
29	78.	94.	8	64	9.	100.	10
30	55.	3.	7	65	17.	82.	5
31	82.	88.	10	66	74.	67.	7
32	73.	28.	6	67	10.	68.	9
33	20.	55.	2	68	48.	19.	6
34	27.	43.	11	69	83.	86.	11
35	95.	86.	9	70	84.	94.	2

Depot coordinates (50,50).

Capacity of vehicle: 50 units.

TABLE XVI

## PROBLEM 5

Site	x	y	Demand	Site	x	y	Demand
1	4.	84.	4	46	75.	95.	11
2	67.	43.	11	47	92.	76.	7
3	16.	57.	10	48	4.	59.	7
4	35.	96.	9	49	100.	10.	6
5	42.	90.	5	50	59.	20.	6
6	40.	86.	8	51	61.	32.	7
7	20.	64.	4	52	3.	65.	4
8	98.	52.	9	53	53.	38.	2
9	50.	92.	10	54	98.	72.	9
10	79.	77.	7	55	66.	72.	11
11	43.	48.	9	56	46.	9.	9
12	49.	53.	3	57	38.	39.	6
13	86.	96.	2	58	77.	95.	8
14	99.	95.	2	59	0.	8.	2
15	1.	92.	6	60	12.	26.	9
16	60.	15.	9	61	3.	52.	5
17	40.	25.	6	62	70.	83.	2
18	84.	47.	9	63	13.	69.	3
19	68.	99.	2	64	92.	77.	6
20	71.	81.	10	65	38.	29.	7
21	56.	98.	6	66	72.	52.	4
22	3.	46.	4	67	4.	61.	8
23	24.	14.	10	68	28.	24.	10
24	95.	69.	9	69	0.	41.	8
25	30.	21.	3	70	14.	57.	9
26	57.	93.	2	71	65.	64.	7
27	87.	41.	7	72	50.	51.	9
28	29.	42.	8	73	29.	76.	9
29	97.	7.	10	74	74.	30.	4
30	93.	56.	2	75	6.	72.	9
31	29.	93.	6	76	36.	17.	9
32	65.	67.	2	77	80.	18.	3
33	55.	59.	9	78	28.	61.	3
34	37.	64.	3	79	70.	34.	5
35	16.	55.	4	80	91.	29.	9
36	95.	64.	10	81	7.	49.	3
37	49.	37.	11	82	22.	47.	1
38	2.	23.	11	83	34.	5.	11
39	30.	75.	7	84	95.	67.	10
40	64.	17.	3	85	32.	11.	4
41	17.	1.	4	86	21.	57.	7
42	44.	20.	5	87	97.	74.	7
43	9.	62.	10	88	61.	52.	4
44	57.	93.	9	89	63.	31.	2
45	2.	82.	10	90	7.	65.	5

Depot coordinates (50,50).

Capacity of vehicle: 50 units.

TABLE XVII

## PROBLEM 6

Site	x	y	Demand	Site	x	y	Demand
1	68.	52.	10	41	95.	31.	11
2	50.	71.	7	42	51.	17.	10
3	72.	40.	5	43	45.	52.	7
4	38.	15.	9	44	74.	91.	2
5	98.	87.	8	45	83.	30.	5
6	63.	55.	3	46	45.	31.	7
7	68.	11.	4	47	14.	77.	2
8	80.	62.	1	48	59.	66.	6
9	99.	99.	7	49	3.	35.	4
10	58.	56.	7	50	20.	83.	8
11	55.	18.	9	51	70.	24.	2
12	18.	47.	10	52	66.	82.	3
13	71.	78.	7	53	13.	99.	7
14	26.	7.	2	54	42.	70.	5
15	63.	17.	6	55	35.	44.	2
16	76.	41.	4	56	88.	47.	8
17	1.	48.	3	57	11.	91.	11
18	40.	27.	1	58	43.	70.	6
19	88.	88.	11	59	1.	78.	6
20	34.	3.	10	60	11.	39.	7
21	47.	99.	7	61	77.	18.	9
22	83.	1.	2	62	3.	61.	7
23	64.	60.	8	63	91.	91.	6
24	16.	1.	2	64	77.	19.	2
25	74.	8.	7	65	77.	3.	6
26	84.	93.	9	66	21.	68.	9
27	97.	65.	9	67	51.	31.	4
28	12.	15.	2	68	28.	90.	4
29	17.	10.	7	69	74.	7.	8
30	30.	90.	4	70	7.	33.	2
31	53.	6.	9	71	51.	100.	6
32	77.	5.	11	72	69.	58.	3
33	83.	52.	4	73	21.	10.	10
34	64.	30.	3	74	74.	84.	9
35	65.	98.	6	75	63.	3.	5
36	86.	26.	10	76	56.	18.	4
37	38.	12.	11	77	96.	50.	11
38	37.	28.	4	78	64.	65.	7
39	44.	99.	4	79	34.	56.	4
40	27.	78.	5	80	79.	95.	6

TABLE XVII (Continued)

Site	x	y	Demand	Site	x	y	Demand
81	33.	10.	3	116	79.	5.	7
82	18.	54.	9	117	64.	96.	8
83	38.	40.	2	118	95.	90.	4
84	97.	89.	10	119	1.	93.	8
85	82.	22.	6	120	74.	82.	11
86	78.	60.	9	121	1.	4.	4
87	32.	81.	5	122	15.	13.	2
88	23.	15.	11	123	46.	91.	3
89	14.	0.	9	124	71.	48.	8
90	26.	16.	8	125	54.	34.	2
91	60.	51.	9	126	80.	6.	6
92	94.	24.	4	127	47.	24.	9
93	92.	77.	3	128	54.	6.	11
94	22.	72.	11	129	78.	66.	11
95	12.	35.	10	130	8.	18.	3
96	84.	25.	3	131	3.	64.	11
97	33.	98.	5	132	74.	79.	2
98	38.	84.	10	133	9.	77.	11
99	53.	62.	2	134	37.	65.	11
100	35.	92.	3	135	39.	92.	8
101	29.	84.	7	136	0.	29.	9
102	23.	89.	11	137	65.	4.	7
103	8.	64.	4	138	29.	78.	11
104	95.	21.	6	139	97.	77.	5
105	38.	29.	8	140	4.	56.	11
106	77.	48.	9	141	29.	27.	3
107	36.	31.	10	142	30.	3.	7
108	3.	66.	9	143	83.	48.	5
109	46.	90.	7	144	44.	25.	5
110	35.	20.	4	145	39.	17.	3
111	29.	35.	9	146	46.	97.	9
112	9.	61.	11	147	38.	39.	5
113	64.	13.	10	148	18.	34.	3
114	63.	39.	8	149	85.	62.	9
115	77.	25.	3	150	41.	70.	8

Depot coordinates (50,50).  
Capacity of vehicle: 50 units.

## APPENDIX D

### USER INSTRUCTIONS FOR INTERACTION PHASE

Once the results of the Clarke and Wright-Lin program are received by an analyst, changes may be desired. The changes may include a different sequencing of nodes, the creation of a new route, or some other system perturbation. Once a change is decided upon, the new information must be fed into the computer. The data outlined in Appendix A must be read in with one alteration. The first data card, which contains the number of stops (NSTOP) and vehicle capacity (NCAP), also contains a flag variable, MAN, which tells the computer whether or not the interaction phase is being used. If  $MAN = 0$ , as in Appendix A, the interaction phase is not invoked. On the other hand,  $MAN > 0$  means an analyst is reading in a new solution.

After the data of Appendix A is fed into the computer, an analyst must read in information about the new solution. The first data card contains the number of routes (NXTRT) in the proposed solution. NXTRT is read according to an I10 format using the first ten columns. The remaining data cards contain the node sequence of the proposed routes. Each card associated with a new route contains the number of nodes (N) on the proposed route. N is read according to an I3 format in the second, third and fourth columns. The remaining columns contain the node sequence of the new route. Each node is read into the array IX under the format 19I4. Should there be more than nineteen nodes on a

route, the node sequence is continued on the following data card(s) using the 20I4 format.

#### Example

For the eight node problem of Appendix A, the Clarke and Wright-Lin program generates the following solution:

9 - 2 - 8 - 7 - 9

9 - 4 - 5 - 3 - 9

9 - 1 - 9

9 - 6 - 9

An analyst is interested in seeing the effect of adding node 6 to the first route following node 8. Table XVIII shows the required data cards for use of the interaction phase. Notice that card one is the same as in Appendix A except for the 1 in column thirty. This tells the computer the analyst is proposing his own solution. Cards two through ten are identical to the ones of Appendix A. Card eleven is the first card of the proposed solution. It contains the number of proposed routes (NXTRT=3). Card twelve consists of the number of nodes on the first route (N=4) and the node sequence (2-8-6-7). Likewise, cards thirteen and fourteen contain the data for routes two and three. It should be pointed out that since routes two and three are not changed they need not be included in the interaction phase. If some nodes are excluded from the interaction phase, the number of nodes in the system, NSTOP, must be reduced accordingly.



VITA

Jeffrey Allan Robbins

Candidate for the Degree of

Master of Science

Thesis: A PROGRAM FOR SOLUTION OF LARGE SCALE VEHICLE ROUTING PROBLEMS

Major Field: Industrial Engineering and Management

Biographical:

Personal Data: Born in Richmond, Virginia, July 10, 1952, the son of Mr. and Mrs. S. Robbins.

Education: Graduated from Highland Springs High School, Highland Springs, Virginia, in June, 1970; received Bachelor of Science degree in Statistics from Virginia Polytechnic Institute and State University in June, 1974; completed requirements for Master of Science degree at Oklahoma State University in May, 1976.

Professional Experience: Operations Research Analyst, Federal Power Commission, 1974-75; graduate research assistant, Oklahoma State University, Center for Local Government Technology, 1974-76.