

WORKLOAD BALANCING IN VEHICLE

ROUTING PROBLEMS

By

JERRY DENVER ALLISON

Bachelor of Science
University of Texas at Arlington
Arlington, Texas
1968

Master of Engineering
Texas A and M University
College Station, Texas
1970

Submitted to the Faculty of the Graduate College
of the Oklahoma State University
in partial fulfillment of the requirements
for the Degree of
DOCTOR OF PHILOSOPHY
December, 1986

Thesis
1986 D
A438W
cop. 2



WORKLOAD BALANCING IN VEHICLE

ROUTING PROBLEMS

Thesis Approved:

Marvin Palmer Terrell

Thesis Adviser

Joe H. Mize

Allen C. Schaeffer

Michael A Branson

Donald W. Grace

Norman N. Durham

Dean of the Graduate College

PREFACE

This study is concerned with the vehicle routing problem (VRP) in which it is desired to minimize, in addition to the total distance over all routes, the deviation in workload among the routes. Two workload elements are considered: (1) the total distance or time spent driving, and (2) the total weight or amount of goods delivered. This problem is termed the workload balanced vehicle routing problem (WBVRP). The purpose of the study is to develop an interactive model to solve the WBVRP using multiple criteria analysis.

I wish to express my gratitude to my major adviser and chairman of my Ph.D. committee, Dr. M. Palmer Terrell, for his encouragement and guidance during this study and throughout my doctoral program. I wish also to thank the members of my committee, Dr. Michael H. Branson, Dr. Joe H. Mize, Dr. Allen C. Schuermann, and Dr. Donald W. Grace, for their interest and assistance.

I wish also to thank the staff of the University Computer Center for their help and cooperation in developing the graphics capability for the interactive program.

The love, understanding, and patience of my wife, Susan, have enabled me to complete this dissertation. Her faith and trust, and that of my two daughters, Ginger and Leslie, have helped me endure the many long and difficult hours required for this research.

TABLE OF CONTENTS

Chapter	Page
I. INTRODUCTION	1
Problem Definition	1
Advantages of Workload Balancing	6
Measures of Workload Imbalance	7
Goals and Objectives of Research	9
Research Outline	10
II. LITERATURE REVIEW	11
The Traveling Salesman Problem	11
Exact Procedures	12
Heuristic Procedures	15
The Vehicle Routing Problem	27
Definition	27
Classification	28
Solution Techniques	30
Exact Methods	31
Cluster First - Route Second	33
Route First - Cluster Second	34
Savings/Insertion	35
Improvement/Exchange	39
Mathematical-Programming Based	40
Interactive	43
Workload Balancing in VRPs	46
Summary	51
III. GENERAL MODEL STRUCTURE	54
Assumptions	54
Mathematical Representation of the WBVRP	55
Multiple Criteria Optimization	57
Terminology	57
Methods	58
The Method of Satisfactory Goals	62
The Method of Satisfactory Goals Applied to the WBVRP	64
Summary	69
IV. SINGLE-OBJECTIVE ALGORITHMS	72
Minimization of Total Distance	72
Minimum Distance Algorithm	72
Effectiveness of Algorithm	78

Chapter	Page
Efficiency of Algorithm	79
Minimization of Route-Length Deviation	81
Route-Length Deviation Algorithm	81
Effectiveness of Algorithm	89
Efficiency of Algorithm	90
Minimization of Route-Load Deviation	95
Route-Load Deviation Algorithm	95
Effectiveness of Algorithm	97
Efficiency of Algorithm	100
Summary	104
V. INTERACTIVE COMPUTER PROGRAM	105
Introduction	105
Program Description	106
Menu	106
Multiple Runs of Single-Objective Algorithms	108
Goal Tradeoffs	110
Flow Chart	114
Evaluation of Interactive Computer Program	116
Effectiveness of Program	116
Convergence Analysis	116
Nondominance Analysis	125
Efficiency of Program	125
Example Problem	126
Summary	137
VI. WORKLOAD BALANCING COSTS	138
Introduction	138
Method of Analysis	139
Results	147
Depot Location Penalty	147
Workload Balancing Penalties	151
Route-Load Balancing	152
Route-Length Balancing	152
Route-Load and Route-Length Balancing	153
Conclusions	153
VII. SUMMARY, CONCLUSIONS, AND RECOMMENDATIONS	161
Summary	161
Conclusions	164
Recommendations for Further Research	165
BIBLIOGRAPHY	168
APPENDIXES	175
APPENDIX A - APPROXIMATE NUMBER OF QUALIFIED ARC COMBINATIONS IN DEVIATION MINIMIZATION ALGORITHMS	176

Chapter	Page
APPENDIX B - VEHICLE ROUTING PROBLEM DATA	179
APPENDIX C - DATA FOR REGRESSION MODELS	187
APPENDIX D - FORTRAN PROGRAM LISTING	192

LIST OF TABLES

Table	Page
2.1. VRP Algorithms	53
4.1. Effectiveness of Distance Minimization Algorithm	80
4.2. Regression Table for 3-opt Solution Times	82
4.3. Effectiveness of Route-Length Deviation Algorithm	91
4.4. Regression Table for Solution Time of Route-Length Deviation Algorithm as a Function of Number of TSPs Solved	92
4.5. Regression Table for Solution Time of Route-Length Deviation Algorithm as a Function of Problem Size, Number of Routes, Workload Imbalance, and Achievement Level Relaxation	96
4.6. Effectiveness of Route-Load Deviation Algorithm	99
4.7. Regression Table for Solution Time of Route-Load Deviation Algorithm as a Function of Number of TSPs Solved	102
4.8. Regression Table for Solution Time of Route-Load Deviation Algorithm as a Function of Problem Size, Number of Routes, and Workload Imbalance	103
5.1. Actual Versus Promised Improvement in Objective Function Provided by Tradeoffs	113
5.2. Convergence of Interactive Computer Program Using Different Initial Points	124
6.1. Customer Demand Patterns	141
7.1. Solution Times and Solution Quality of Single-Objective Algorithms	163
B.1. Gaskell's 22-City Problem	180
B.2. Gaskell's 29-City Problem	181
B.3. Gaskell's 32-City Problem	182
B.4. Christofides and Eilon's 50-City Problem	183

Table	Page
B.5. Christofides and Eilon's 75-City Problem	184
B.6. Christofides and Eilon's 100-City Problem	185
B.7. Example 33-City Problem	186
C.1. Data for 3-opt Solution Times Regression Model.	188
C.2. Data for Route-Length Deviation Regression Models	189
C.3. Data for Route-Load Deviation Regression Models	190

LIST OF FIGURES

Figure	Page
1.1. Minimum Distance Solution to 33-City Vehicle Routing Problem	3
1.2. Workload Balanced Solution to 33-City Vehicle Routing Problem	4
2.1. The Savings Method Illustrated	17
2.2. Convex-Hull TSP Method of Wiorkowski and McElvain	21
2.3. Largest-Angle Convex-Hull TSP Method of Norback and Love	22
2.4. Christofides' Heuristic	24
3.1. Method of Satisfactory Goals Illustrated	65
3.2. General WBVRP Model Structure	70
4.1. 2-opt and 3-opt Arc Exchanges	74
4.2. Restructuring Routes Using Artificial Depots	75
4.3. Distance Minimizing Algorithm	77
4.4. Use of Arc Exchanges for Clustering Two Routes	85
4.5. Use of Arc Exchanges for Clustering Three Routes	86
4.6. Route-Length Deviation Algorithm	88
4.7. Route-Load Deviation Algorithm	98
5.1. Screen Display Containing Menu, Problem Status, and Tradeoff Information	109
5.2. Flow Chart for Main Program	117
5.3. Example Problem: Beginning Route Set	127
5.4. Example Problem: Solution Number 2	129
5.5. Example Problem: Solution Number 3	131

Figure	Page
5.6. Example Problem: Solution Number 4	133
5.7. Example Problem: Solution Number 5	135
5.8. Example Problem: Solution Number 6	136
6.1. First Customer Location Pattern	141
6.2. Second Customer Location Pattern	144
6.3. Third Customer Location Pattern	145
6.4. Fourth Customer Location Pattern	146
6.5. Depot Location Penalties for Minimum-Distance Problems	149
6.6. Route-Load Balancing Penalties	154
6.7. Route-Length Balancing Penalties	156
6.8. Penalties for Balancing Both Route Load and Route Length . . .	158

CHAPTER I

INTRODUCTION

Problem Definition

Over the past twenty-five years, the vehicle routing problem (VRP) has received the attention of many researchers. The problem has typically been solved using single-objective optimization or heuristic methods, the objective usually being the minimization of total route time or distance traveled. Problem constraints might vary, but usually include maximum route time or distance constraints, vehicle capacity constraints, and might include limits on the number of available vehicles.

One of the objectives which a distribution manager must usually consider is the equitable distribution of workload among the workers in the system. For purposes of this research, the following elements of workload are considered:

1. Total driving workload, expressed as the total distance driven or total time spent driving by a given crew.
2. Total handling workload, expressed as a total weight of goods picked up or delivered, or total time spent in loading/unloading goods picked up or delivered, or total stops made by a given crew.

The particular units of measure for these workload elements depend upon the specific problem being solved.

The workload-balanced vehicle routing problem (WBVRP), then, is the VRP which has as its objectives, in addition to the minimization of total time or distance traveled, the equitable distribution among the workforce of one or more of the elements above. By this definition, the WBVRP is seen to be a multiobjective problem.

An 'ideal' solution to the WBVRP would achieve a minimum total time or distance traveled, and would have both workload elements distributed equally among the workforce. However, because of the combinatorial nature of the problem and the tradeoffs necessary in achieving a perfectly balanced solution, this 'ideal' solution is usually not possible. Instead, the preference structure of the route planner dictates the tradeoffs among the objectives, and a compromise solution is accepted.

As an example, consider the 33-city VRP represented in Figure 1.1. The five routes shown are a minimum-distance set, the total distance over all routes being 174 miles. Note, however, that the lengths of individual routes vary from 21 miles for the shortest route (route number five) to 43 miles for the longest route (route number four). Also, note that the load carried on the lightest route (route number three) is 1190 pounds, while the load carried on the heaviest route (route number four) is 1460 pounds. A more balanced set of routes for the same VRP is shown in Figure 1.2. In this route set, the difference between the shortest and longest routes (routes three and two, respectively) is only seven miles, and the difference between the lightest and heaviest routes (routes one and four, respectively) is only 90 pounds. Note, however, that the total distance traveled over the routes in Figure 1.2 is 186 miles. The difference of 12 miles in total distance

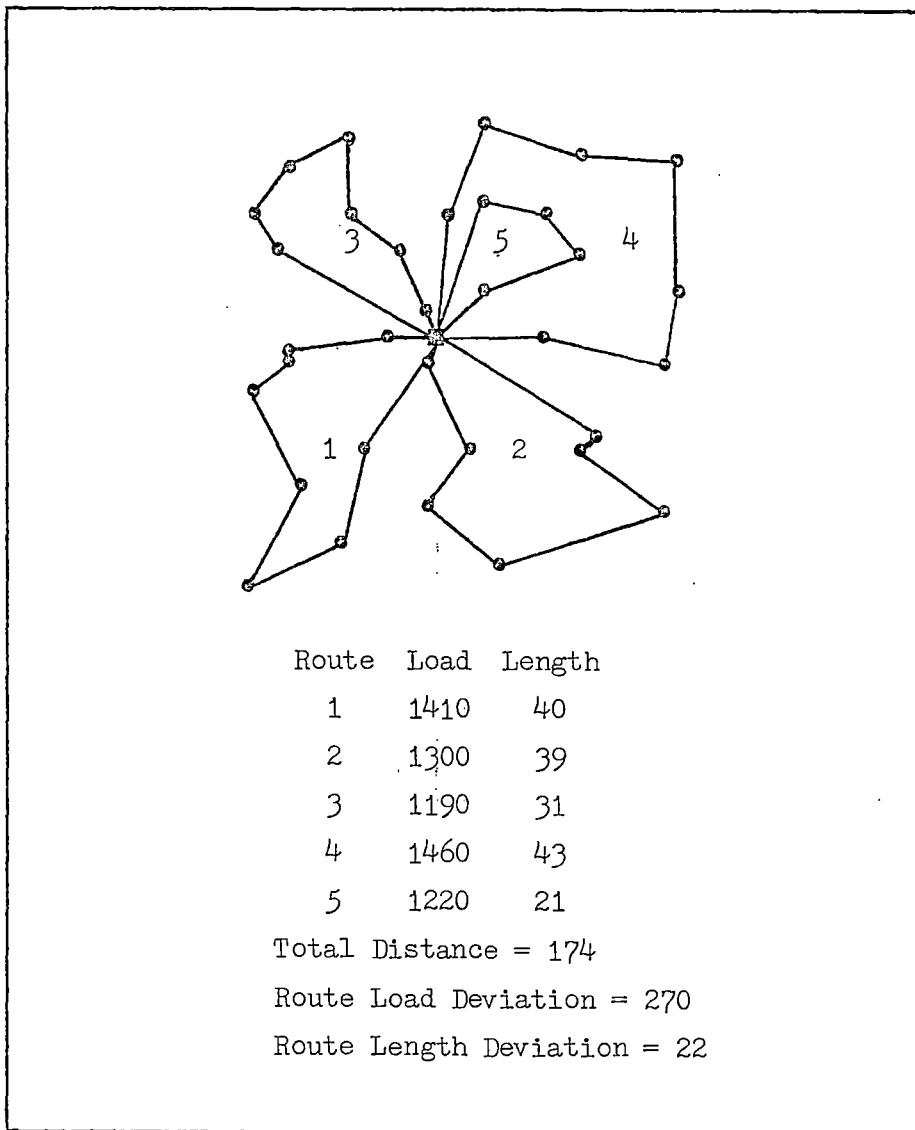


Figure 1.1. Minimum Distance Solution to 33-City Vehicle Routing Problem

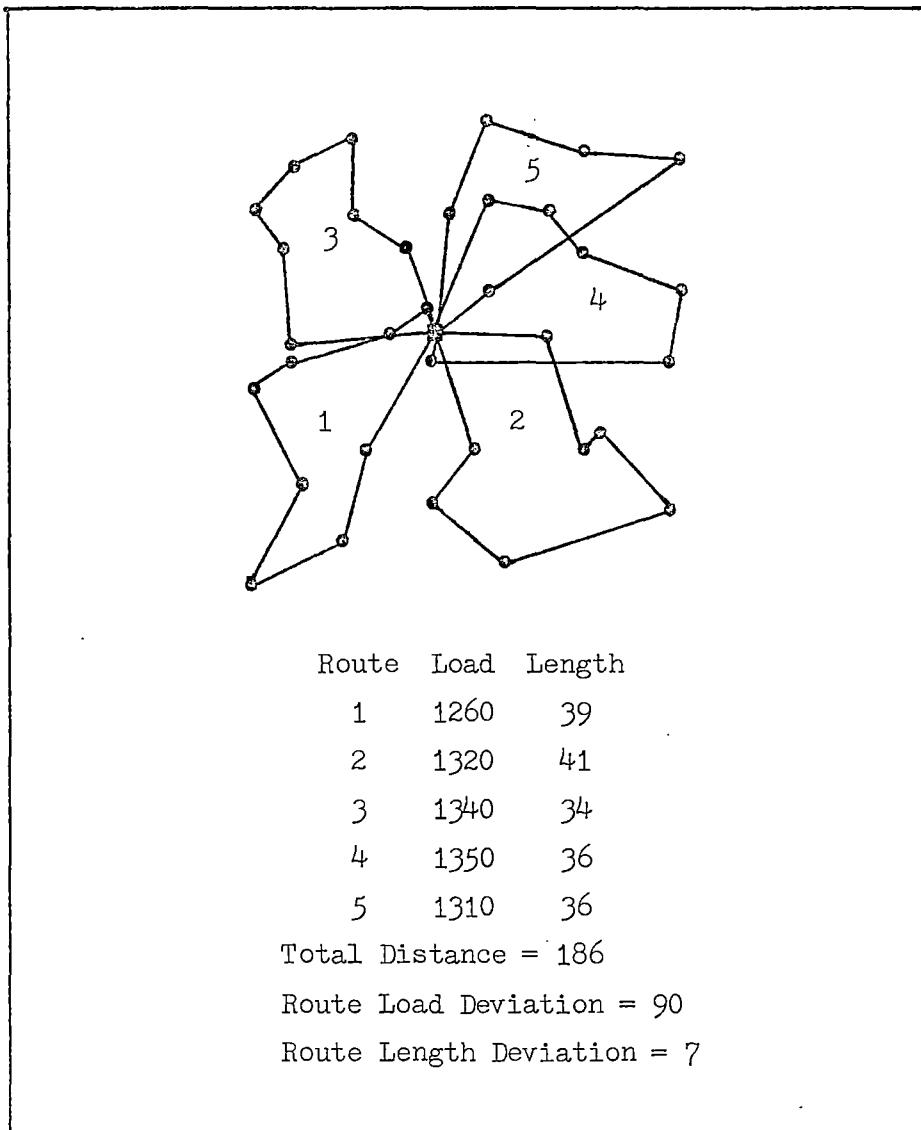


Figure 1.2. Workload Balanced Solution to 33-City Vehicle Routing Problem

between Figures 1.1 and 1.2 represents the penalty which must be paid to achieve the level of balance shown.

The WBVRP can manifest itself in several different ways. Some delivery crews consist of a driver and a helper, the helper's duty being primarily that of loading or unloading the goods at each stop. A set of routes which exhibits a wide variance in the helper's workload, even if the drivers all have approximately the same amount of driving workload, is not an acceptable set, at least to the helpers who must do the greater share of the handling workload. On the other hand, a set of routes which balance the helpers' workload while exhibiting a wide variance in the drivers' workload is not acceptable to some of the drivers. In trying to achieve an acceptable solution, the route planner must weigh the importance of management's objective, usually the minimization of cost, against the competing claims of different sectors of the delivery workforce, and make appropriate tradeoffs.

Even with crews which consist of only a driver, workload balancing is an important issue. Here the driver does all the work, so the route planner does not have different workforce skills to consider. He does, however, have the problem of trading off different levels of the workload elements against each other and against the total distance or time objective. The problem exists regardless of the perceived relative desirability of the different workload elements to the workforce. In one situation, the driving element might be preferred over the handling element, especially if the handling element requires a large amount of heavy lifting and/or carrying effort. In another situation, the handling element might be preferred over the driving element. An example of this would be the case in which adverse driving conditions exist. The

important thing is not the relative desirability of one workload element versus the other, but the fact that preferences do exist, necessitating that tradeoffs be made among them.

Routing problems closely related to the WBVRP are those in which the loads carried by the vehicles do not necessarily represent one of the workload elements above, per se, but in which it is nonetheless important to balance the vehicle loads. An example of this is the VRP in a driver-sell environment, in which at least part of the driver's earnings are dependent upon the quantity of goods delivered. Another example is the school-bus routing problem in which it is desired to equalize to some extent the number of children carried in the different buses, thereby distributing the responsibility for the children's safety among the bus drivers. Because such problems can be formulated in the same manner as the WBVRP, they can be considered to be included in the overall scope of this research.

Advantages of Workload Balancing

A set of workload-balanced routes has several potential advantages over a minimum-distance or minimum-time set of routes. These advantages fall within four areas.

Employee Relations

For most companies, the primary reason for using workload-balanced routes is that the perception of workload equity by the workforce should enhance employee morale. This in turn can lead to fewer worker complaints, lower levels of absenteeism, greater levels of cooperation, and a general trend toward higher labor productivity.

Crew Scheduling

Company and/or union goals sometimes specify certain levels of workload equity over a period of time. If the crews are always assigned to workload-balanced routes, there is no need to resort to sophisticated crew scheduling systems in order to attain these goals.

Route Stability

Fixed routes are sometimes desired. However, if customer demand changes over time, these routes will eventually have to be altered to meet vehicle capacity and route length constraints. A set of balanced routes requires fewer alterations over a long period of time than does a minimum-distance or minimum-time set of routes (Dileepan, 1984).

Fleet Management

Using workload-balanced routes, all vehicles in the fleet undergo approximately the same mileage and vehicle loads. Therefore, many fleet management decisions related to a particular vehicle will also apply to the fleet as a whole. Included are vehicle replacement intervals, preventive maintenance schedules, and intervals for tire rotations, vehicle inspection, oil changes, and so forth.

Measures of Workload Imbalance

There are several measures that could be related to the two workload elements listed on page one. The two measures which are used in this research are discussed below.

Route Length Deviation

Route length deviation is a measure of the first workload element. It is defined as the maximum route length in a route set minus the minimum route length in the set. 'Route length' can be interpreted to mean either distance traveled or time spent in traveling over a route. If applicable, the route lengths include 'drop allowances' at the various stops, measured in the same distance or time units.

Route Load Deviation

Route load deviation, a measure of the second workload element, is defined to be the maximum route load carried in a route set minus the minimum route load carried. 'Route load' is usually thought of as the weight of goods carried on a route. It may, however, be defined as the volume of product, number of items carried, number of stops on a route, or the number of customers carried (e.g., in a bus routing problem).

The route loads correspond to the total demand of all stops on a route.

It could be argued that other measures of workload imbalance are appropriate. For instance, the standard deviation, average deviation, sum of squared deviations, or sum of absolute deviations could be used. Certainly, from a distribution manager's viewpoint, these measures might be adequate. Also, from a computational standpoint, the sum of squared deviation can be easily determined in 'pairwise exchange' heuristics (Dileepan, 1984). However, since employee morale is considered to be one of the primary reasons for using workload-balanced routes, and since the measures defined above should be more meaningful to members of the distribution workforce, those measures are the ones which have been adopted for this research.

Goals and Objectives of Research

Objectives

There are two primary objectives for this research. They are:

1. Examine the use of multicriteria analysis in developing vehicle routes which offer an equitable distribution of workload among the workforce.
2. Determine the implications of workload balancing to distribution management under differing conditions of customer demand and location.

Goals

To meet the objectives above, four specific goals are delineated.

They are:

1. Develop a multiobjective model structure to solve the WBVRP, utilizing user interaction to make tradeoffs among the three objectives of the problem.
2. Develop and evaluate methods to minimize each of the three objective functions of the WBVRP.
3. Incorporate the multiobjective model structure into an interactive computer program, and evaluate the performance of the program in terms of efficiency (solution times) and effectiveness (solution values).
4. Solve different WBVRP problems which vary in their customer demand patterns (i.e., constant demand vs variable demand) and in the relationship between depot and customer locations.

Determine the penalty, in overall time or distance units, which

distribution managements must pay in order to balance the two workload elements under these conditions.

Research Outline

The research into workload balancing in VRPs is contained in the six remaining chapters. Chapter II contains a review of the literature covering the vehicle routing problem and route balancing in VRPs. Chapter III covers the development of an interactive multiobjective model structure, based on the Method of Satisfactory Goals (Benson, 1975), which can be used to solve the WBVRP. In Chapter IV, three different single-objective algorithms, necessary to the implementation of the multiobjective model, are developed and evaluated in terms of their efficiency and effectiveness. Chapter V describes and demonstrates an interactive computer program which is used to implement the multi-objective model, and contains an evaluation of the program in terms of its ability to converge to a solution from different starting points. In Chapter VI, the solution results of several problems which vary in customer demand and location patterns are presented, and implications for distribution management are offered. Finally, Chapter VII contains conclusions from this research and offers suggestions for further research in this area.

CHAPTER II

LITERATURE REVIEW

This chapter contains a review of research in the area of vehicle routing. Because of the importance of the traveling salesman problem (TSP) in many approaches used to solve the vehicle routing problem (VRP), the chapter begins with a review of work done toward solving the TSP. Next, approaches taken to solve the VRP are reviewed. Finally, research efforts in the area of workload-balanced vehicle routes are covered.

The Traveling Salesman Problem

The TSP, generally credited to Professor Hassler Whitney of Princeton University in 1934 (Flood, 1956), can be stated as follows: A salesman wishes to leave his home and visit each of $n-1$ cities only once, then return. If the cost of traveling between city i and city j is c_{ij} , the salesman wishes to minimize the sum of the travel costs, $\sum c_{ij}$. If $c_{ij}=c_{ji}$ for all i and j , the problem is said to be symmetric; otherwise, it is said to be asymmetric. The TSP is one of the most extensively researched problems in operations research. Approaches to solving it have been surveyed by Bellmore and Nemhauser (1968), Eilon et al., (1971), Christofides (1979), and Lawrence (1981).

The TSP is NP-complete. As such, there are no known algorithms which will obtain an exact solution to the problem in polynomial time. For large problems, therefore, a heuristic approach is usually taken. In

the following paragraphs, both approaches are presented. Exact solution procedures are covered first, followed by heuristic procedures.

Exact Procedures

Dynamic Programming. Both Bellman (1962) and Held and Karp (1962) applied the principle of optimality to solve the TSP using dynamic programming. The major difficulty with this approach is that the storage requirements for the recursive relationships grow exponentially as the number of cities in the problem is increased. Held and Karp solved a 13-city asymmetric problem, which was the limit of their computing capacity (an IBM 7090 with 32K memory). However, Bellmore and Nemhauser (1968) demonstrated that a machine of this size could solve an 18-city problem by using auxiliary storage and a judicious selection of the values to be maintained in memory.

Integer Programming. Dantzig, Fulkerson, and Johnson (1954) solved a 42-city TSP by using a linear programming formulation. Their method avoided the large number of loop constraints (necessary to prevent subtours) by beginning with only a limited number of constraints and then adding additional ones as necessary. Fractional solutions were eliminated by applying combinatorial arguments on an ad-hoc basis. Miller, Tucker, and Zemlin (1960) used a similar approach but employed Gomory's cutting plane algorithm instead of the combinatorial arguments of Dantzig et al. Martin (1963) solved the 42-city problem using cutting planes and a different set of loop constraints which proved to be efficient in eliminating fractional solutions as well. Miliotis (1976) reported solving symmetric problems involving up to 64 cities by employing an algorithm which first achieved integrality through cutting

planes, then added loop constraints as necessary.

Branch and Bound. The majority of exact solution approaches to the TSP have utilized some form of branch-and-bound procedure. There are two general approaches: (1) tour building and (2) subtour elimination. The first of these, the tour-building approach, was developed by Little et al. (1963), and used a penalty method for determining the branching process. Incidentally, their article marked the first appearance of the expression "branch and bound" in the literature.

The second approach, subtour elimination, was developed by Eastman (1958) and modified by Shapiro (1966) and Bellmore and Malone (1971). Solving an assignment problem provides an initial lower bound and, if no subtours are present, the optimal solution. If subtours are present, then changes to the cost matrix are made to prevent them from occurring in further solutions to the assignment problem.

Christofides (1972) has shown that the tightness of the lower bounds in a branch-and-bound scheme is of more importance than the branching process in determining the effectiveness of the method. In order to improve the bounds over those obtained by previous branch-and-bound procedures, he developed an algorithm which utilizes two transformations of the original cost matrix: (1) a "contraction", which is defined as the replacement of a subtour by a single node, and (2) a "compression", which is defined as the transformation of a matrix which does not satisfy the triangularity condition of metric space into one that does. The triangularity condition is met if the distance between two points is not greater than the distance between the same two points while passing through an intermediate third point. His algorithm proceeds as follows:

Step 1. Set matrix M equal to initial distance matrix. Set lower bound L equal to zero.

Step 2. If triangularity condition is met, go to Step 3. Otherwise, COMPRESS M.

Step 3. Solve assignment problem using matrix M and increase L by this amount.

Step 4. CONTRACT matrix M by replacing subtours by a single node.

Step 5. If matrix M is now 1 x 1, go to Step 6. Otherwise, go to Step 2.

Step 6. End. Lower bound = L.

Using this procedure, bounds which were on the average within 4.7 percent of optimality for symmetric TSPs and within 3.8 percent for asymmetric TSPs were obtained at an average computation premium of only 9 percent increase in time over the original assignment problem.

Balas and Christofides (1981) used the assignment-problem approach to calculating lower bounds in a subtour-elimination scheme, but their method involves the introduction of violated subtour-elimination constraints into the objective function via Lagrangean relaxation techniques. Their approach has been found to provide extremely tight bounds (within one-half percent of the TSP optimum) for asymmetric TSPs. Balas and Christofides have used this Lagrangean approach to solve asymmetric problems of up to 325 cities optimally.

Held and Karp (1970, 1971) developed a Lagrangean relaxation approach to the symmetric TSP involving bounds from minimum spanning trees (1-trees in particular). A minimum 1-tree is comprised of the minimum spanning tree through vertices 2, 3, 4, . . . , n plus the two

minimum-weight arcs connecting vertex 1 to the remainder. If the minimum 1-tree is a tour, then the tour is a solution to the TSP. If the minimum 1-tree is not a tour, it does provide a lower bound on the TSP solution.

The minimum 1-tree problem can be formulated as a relaxation to the TSP. Then, by including those constraints directly in the (1-tree) objective function with their associated Lagrange multipliers, a lower bound is obtained. The greatest such bound, obtained by using the best values of the multipliers, is used in the branch-and-bound procedure. Held and Karp (1971) used an iterative subgradient optimization procedure to determine the optimal values of the multipliers, and Hansen and Krarup (1974) later presented improved methods for doing so. Using this bounding technique, symmetric problems of up to 100 cities (Christofides, 1979) have been solved.

Heuristic Procedures

Because the TSP is NP-complete, problems of large size are usually solved by a heuristic approach. These approximate methods can generally be placed into one of three different categories: (1) tour-building heuristics, (2) tour-improvement heuristics, and (3) composite heuristics. A survey of approximate algorithms for the TSP is contained in Golden et al. (1980).

Tour-building Heuristics. The most common method of building up a complete tour is through the "savings" approach. Clarke and Wright (1964) introduced this approach for the vehicle routing problem (VRP); however, if vehicles are assumed to have infinite capacity, then the resulting solution for the VRP will contain a single route (vehicle), and the solution is valid for the TSP.

The algorithm begins by linking $n-1$ nodes to any single arbitrary node. For convenience, call this node 1. The initial solution will therefore consist of $n-1$ separate subtours, each costing $c_{1i} + c_{i1}$. Now, connecting any two nodes i and j ($i, j \neq 1$) will result in the savings

$$S_{ij} = c_{1i} + c_{1j} - c_{ij} \quad (1.1)$$

if the problem is symmetric, and

$$S_{ij} = c_{il} + c_{lj} - c_{ij} \quad (1.2)$$

if the problem is asymmetric. These savings are sorted in decreasing order, and subtours are formed by going down the savings list and linking the appropriate nodes i and j . Any two nodes can be linked if they are not both already in the same subtour, and if both are linked directly to node 1. Figure 2.1 illustrates the procedure. Golden et al. (1980) have shown the Clarke and Wright procedure to require on the order of $n^2 \lg(n)$ computations, where $\lg(n)$ is the logarithm of n with base 2. Several modifications to the Clarke and Wright procedure have been made by others, but a discussion of these is postponed until the vehicle routing problem (VRP) is discussed. As stated previously, the Clarke and Wright algorithm was not originally developed for the TSP, but for the VRP. It is included here only because some methods, such as the tour-improvement algorithms, require an initial tour to begin with, and the savings algorithm is commonly employed for this purpose.

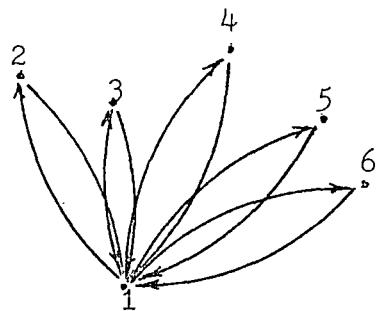
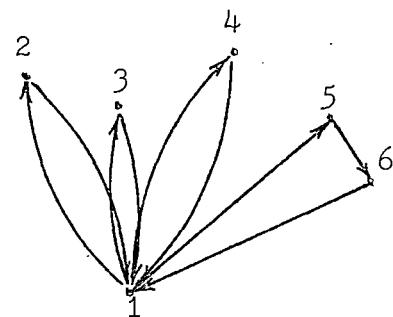
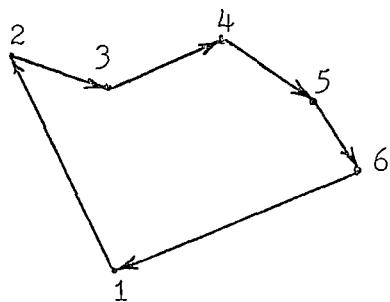
Another class of tour-building heuristics are the "sequential" approaches. The simplest of these is the "proximity" or "nearest neighbor" heuristic (Rosenkrantz et al., 1974). This method begins with any city in the problem, and connects it to the city nearest to it to

	1	2	3	4	5	6
1	∞	4	3	6	7	7
2	4	∞	2	5	7	8
3	3	2	∞	3	5	6
4	6	5	3	∞	2	4
5	7	7	5	2	∞	2
6	7	8	6	4	2	∞

(A) Distance Matrix

$$\begin{aligned}
 S_{23} &= 4 + 3 - 2 = 5 \\
 S_{24} &= 4 + 6 - 5 = 5 \\
 S_{25} &= 4 + 7 - 7 = 4 \\
 S_{26} &= 4 + 7 - 8 = 3 \\
 S_{34} &= 3 + 6 - 3 = 6 \\
 S_{35} &= 3 + 7 - 5 = 5 \\
 S_{36} &= 3 + 7 - 6 = 4 \\
 S_{45} &= 6 + 7 - 2 = 11 \\
 S_{46} &= 6 + 7 - 4 = 9 \\
 S_{56} &= 7 + 7 - 2 = 12
 \end{aligned}$$

(B) Savings Calculations

(C) Initial $n-1$ Tours(D) $n-2$ Tours After S_{56} Applied

(E) Final Tour

Figure 2.1. The Savings Method Illustrated

form a link. Next, the city nearest the last city added is joined to the link. The method continues in this manner until all cities have been included in the tour. Then the first and last cities are joined together. This heuristic requires on the order of n^2 computations (Golden et al., 1980).

Webb (1971) compared the results of the proximity approach with four other sequential heuristics which are based on the idea of a "loss" function. Suppose, during the process of building a tour sequentially, it is desired to link a given unlinked city to its two nearest neighbors (excluding any which have already been linked twice). If the nearest city is not linked to the given city, then the minimum extra distance, or loss, to be paid for not doing so would be at least the extra distance necessary to link it to the third nearest city. In building the tour, links are formed in decreasing sequence of the losses which would be incurred if they are not formed.

The loss function thus described is known as "simple distance loss 1". Problems arise with this loss function in three situations:

1. Cities at both ends of the same chain are among the three nearest neighbors to the given city.
2. Cities at both ends of the same chain have the same unlinked city or cities occurring in the two cities nearest them (converse of situation 1).
3. Either or both cities at the end(s) of a chain occur in the two nearest cities to the cities at both ends of a chain.

To overcome these problems, Webb developed a "simple distance loss 2" function and included a FORTRAN routine for its use, but did not give details of its logic. In a series of 500-city problems, this loss 2

function consistently outperformed the loss 1 function. Using a simpler form of the loss 2 function, which does not require losses to be recalculated after links are formed, problems up to 2500 cities were solved in an average cpu time of 49 seconds on a CDC 6600.

Several "insertion" heuristics have been used in tour-building algorithms. The first of these is the "cheapest insertion" heuristic (Karg and Thompson, 1964). This procedure is as follows:

Step 1. Choose 2 cities to form a partial tour of length 2.

Step 2. Find the city k not in the tour, and the cities i and j in the tour, such that $c_{ik} + c_{kj} - c_{ij}$ is minimized. Insert k between i and j .

Step 3. If the tour is complete, end. Otherwise, go to

Step 2.

Golden et al. (1980) have shown that this procedure requires on the order of $n^2 \lg(n)$ computations.

Another insertion method, the "nearest insertion" heuristic (Rosenkrantz et al., 1974) proceeds as follows:

Step 1. Begin with a subtour consisting of city i only.

Step 2. Find city k such that c_{ik} is minimal, and link the two cities together.

Step 3. Find city k not in the subtour closest to any city in the subtour.

Step 4. Find the cities i and j in the subtour such that $c_{ik} + c_{kj} - c_{ij}$ is minimal. Insert k between i and j .

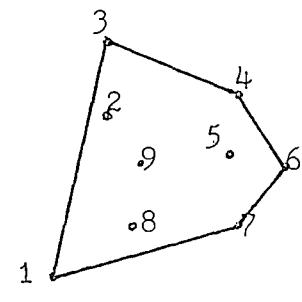
Step 5. If the tour is complete, end. Otherwise, go the Step 3.

Golden et al. (1980) have shown this heuristic to require on the order of n^2 computations.

A "farthest insertion" heuristic (Rosenkrantz et al., 1974) proceeds just as the nearest insertion heuristic above, except that the words "closest to" are replaced by the words "farthest from" in Step 3. This procedure also requires on the order of n^2 computations.

An "arbitrary insertion" heuristic (Rosenkrantz et al., 1974) proceeds as in the nearest insertion heuristic above, except that the city k in Step 3 is chosen arbitrarily. Golden et al. (1980) have shown that this heuristic also requires on the order of n^2 computations.

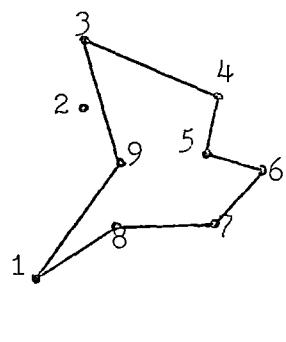
Other insertion tour-building heuristics have been proposed which take advantage of the fact that, for Euclidean problems, the relative ordering of the vertices in the convex hull of the graph remains unchanged in the solution to the TSP (Eilon et al., 1971). Wiorkowski and McElvain (1975) developed a convex-hull procedure which augments an initial extreme-point convex set by iteratively calculating the closest line segments to each free (unassigned) point and assigns new line segments to those points, based on a savings criterion. Refer to Figure 2.2. Norback and Love (1977) inserted new nodes into the convex hull based on the angle formed by the end points of the line segments with the free node at the vertex of the angle. At each step, the free node having the largest angle is inserted between the end points of the line segment, then the angles are recalculated for the next step. For an illustration, see Figure 2.3. Norback and Love (1977) also proposed a method in which new nodes are inserted based on the eccentricity of ellipses formed using the end points of the existing line segments as foci and considering the free nodes to lie on the ellipse thus formed.



(A) Convex Hull (Partial Tour)

<u>Line Segment</u>	<u>Attraction Set</u>
L_{13}	2, 9
L_{34}	\emptyset
L_{46}	5
L_{67}	\emptyset
L_{71}	8

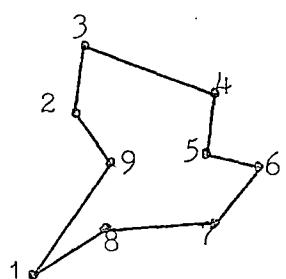
(B) Closest Points to Convex Hull



(C) Partial Tour After Savings Applied

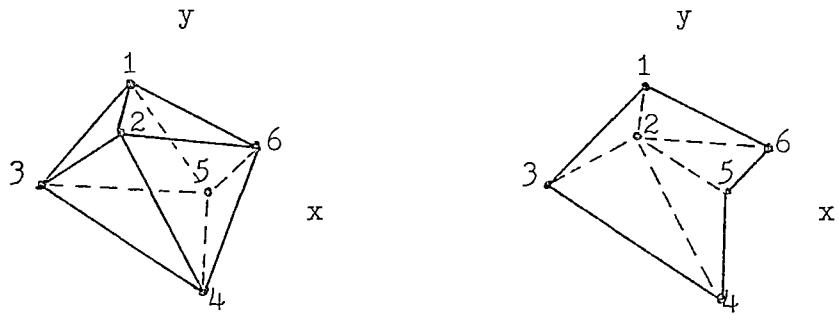
<u>Line Segment</u>	<u>Attraction Set</u>
L_{19}	\emptyset
L_{93}	2
L_{34}	\emptyset
L_{45}	\emptyset
L_{67}	\emptyset
L_{78}	\emptyset
L_{71}	\emptyset

(D) Closest Points to Partial Tour



(E) Final TSP Solution

Figure 2.2. Convex-Hull TSP Method of Wiorkowski and McElvain



- (A) Angles with vertices at interior points. $\angle 456$ is largest. New partial tour is 1-3-4-5-6-1.
- (B) Angles with vertices at remaining interior points. Choose largest one, $\angle 123$ - The tour is 1-2-3-4-5-6-1.

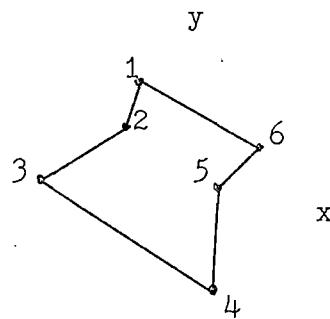


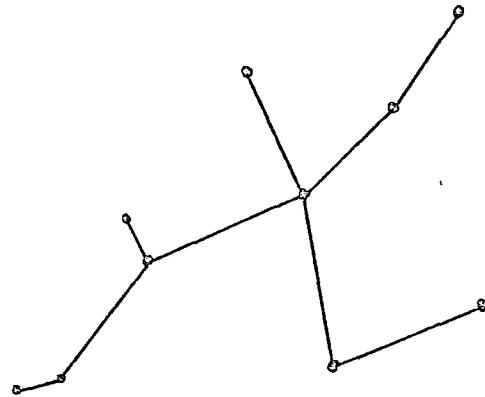
Figure 2.3. Largest-Angle Convex-Hull TSP Method of Norback and Love

Source: Norback and Love (1977).

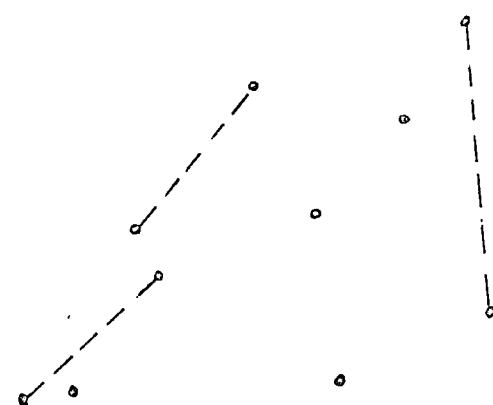
The free node/line segment combination forming the most eccentric ellipse causes the free node to be inserted between the end points. Stewart (1981) presented an algorithm which begins with an initial convex hull and calculates the least cost of inserting each free node k between each pair of line segment end points (i, j) . From all (i, j, k) combinations thus obtained, the combination with the minimum ratio $(c_{ik} + c_{kj})/c_{ij}$ is chosen and node k is inserted between i and j . This process is repeated until a complete tour is obtained.

Christofides' heuristic (1979) is a tour-building technique which can be used to solve the TSP using spanning trees and 1-matching. From the minimum spanning tree through the original set of cities, those vertices having odd degree are chosen (if none exist, the problem is solved). Next, the 1-matching problem is solved for this odd-degreed set. The arcs obtained in this matching solution are added to the original arcs in the spanning tree, resulting in a graph containing all even-degreed vertices. Since the vertices are even-degreed, an Euler's tour can be identified through the augmented set of arcs. The Euler's tour is converted to a Hamiltonian circuit by following the Euler's tour (vertex to vertex), adding each vertex in turn to the Hamiltonian circuit. If a vertex is encountered which has already been added to the Hamiltonian circuit, it is skipped. The resulting Hamiltonian circuit is used as the solution to the TSP. Figure 2.4 illustrates Christofides' heuristic.

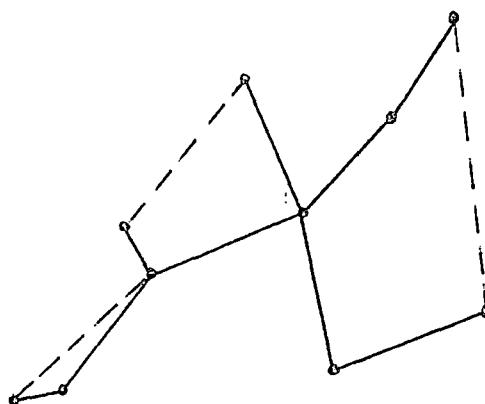
Tour-improvement Heuristics. Given an initial solution to the TSP (from one of the methods of the previous section, for example), the methods of this section attempt to improve that solution through various means. One of the earlier such approaches is due to Croes (1958). For



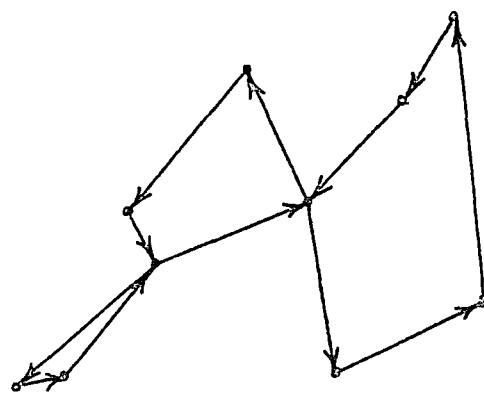
(A) Minimum Spanning Tree



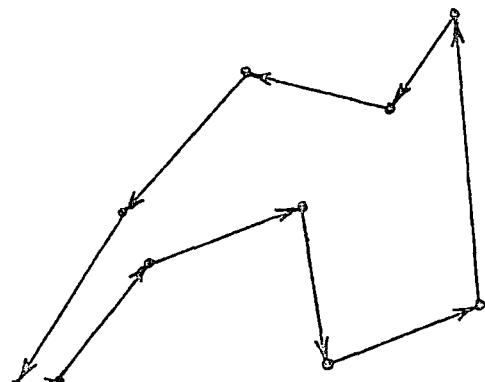
(B) 1-Matching on Odd-Degreed Nodes



(C) Union of Arcs from Minimum Spanning Tree and 1-Matching



(D) Euler's Tour



(E) Hamiltonian Circuit (TSP Solution)

Figure 2.4. Christofides' Heuristic

symmetric problems, it is possible to transform an initial solution through a series of "inversions". These inversions are simple exchanges of arc pairs which yield reductions in the overall travel cost by producing intersectionless routes (routes which do not cross themselves).

Reiter and Sherman (1965) and Lin (1965) independently developed branch-exchange heuristics designed to provide improved solutions to initial tours. The terminology due to Lin will be used here. A tour is said to be k-optimal (or k-opt) if it is impossible to improve the tour by replacing any k of its arcs by any other set of k arcs. By this definition, the intersectionless tours of Croes are seen to be 2-opt. Any tour which is k-opt is also (k-1)-opt. This technique requires on the order of n^k calculations (Golden et al., 1980). Lin and Kernighan (1973) extended the approach to dynamically determine the value of k, based on a set of stopping rules, instead of specifying it in advance.

The original k-opt method as presented by Lin (1965) requires the examination of approximately $\binom{n}{k}(k-1)!2^{k-1}$ combinations of replacement arcs in order to ensure k-optimality (Eilon et al., 1971). This is true because all possible arcs, regardless of their lengths, are examined as possible replacements. However, there are many arc combinations which cannot possibly reduce the overall tour length because of their total length compared with the length of the arcs they are intended to replace. Christofides and Eilon (1972) developed a k-opt procedure which eliminates from consideration all such unrewarding combinations. In a 100-city problem, for example, the number of combinations which were examined to prove 3-optimality was only 18,000 as compared with approximately one million for the original 3-opt method. This advantage

becomes even more significant for larger problems.

Stewart (1985) has taken a different approach toward reducing the computational effort required in achieving 3-optimality. This approach requires the calculation of J minimum spanning trees in the following manner:

Step 1. From the original graph, calculate minimum spanning tree. Remove arcs in solution, forming new graph G .

Set $I = 1$.

Step 2. From G , calculate minimum spanning tree. Remove arcs in solution, forming new graph G . Set $I = I+1$. If $I = J$, go to Step 3. Else, repeat Step 2.

Step 3. From all arcs in the J spanning trees, select an initial tour.

Step 4. Apply 3-opt procedure, allowing only those arcs in the J spanning trees to be used in exchanges.

Although the computational effort is significantly reduced using this approach, the following points should be noted: (1) A tour might not be found in Step 3. If not, it is possible to obtain a tour by using some arcs not found in the J spanning trees. (2) 3-optimality is not guaranteed by this method, since some of the arcs excluded by the algorithm could conceivably improve the solution.

Composite Heuristics. Golden et al. (1980) investigated several procedures which they terms "composite" methods. The basic composite procedure consists of the following three steps:

Step 1. Use a tour-building heuristic to obtain an initial tour.

Step 2. Apply a 2-opt procedure to the tour found in Step 1.

Step 3. Apply a 3-opt procedure to the tour found in Step 2.

Several variations to the basic procedure exist. In an experiment consisting of several 100-city problems, Golden et al. (1980) found that the basic composite procedure given above will find a solution within two to three percent of optimality in most cases. To bring this figure down to one to two percent, the basic procedure must be repeated several times. The consistently best procedure was found to be one in which either an arbitrary-insertion or farthest-insertion heuristic was used in Step 1, the entire composite procedure being repeated ten times. Computation times are not reported.

The Vehicle Routing Problem

Definition

The vehicle routing problem (VRP) was first proposed by Dantzig and Ramser (1959) as a "truck dispatching problem." Since that time, many modifications to the problem definition (and solution techniques) have been made. Generally speaking, however, the basic VRP can be defined as the problem of generating an efficient set of routes from a central depot to a number of customers, each having a known demand, without violating vehicle capacity constraints or individual route time-or-distance constraints. The VRP has been surveyed by Pierce (1969), Turner et al. (1974), Mole (1979), Lawrence (1981), and Bodin et al. (1983).

The routing problem is sometimes known as the "delivery" problem. However, it should be clear that demand points can involve either pickups or deliveries (some formulations involve both). Typical applications are newspaper delivery, garbage collection, mail delivery, electric meter reading, milk distribution, industrial gas distribution, school-bus routing, etc.

A problem related to the VRP is the vehicle scheduling problem (VSP). Whereas the VRP is basically a spatial problem without time constraints (other than the possible constraint on route duration), the VSP is both a spatial and temporal problem¹. An example of temporal constraints would be a set of time windows during which the various deliveries or pickups must be made. This category of problems is not covered in this review, with the exception of certain scheduling methods which contain significant contributions to the routing problem.

Classification

Vehicle routing problems can be classified according to several characteristics. The following taxonomy for vehicle routing and scheduling problems is taken from Bodin and Golden (1981):

- A. Time to service a particular node or arc
 - 1. time specified and fixed in advance (pure scheduling problem)
 - 2. time windows (combined vehicle routing and scheduling problem)
 - 3. time unspecified (vehicle routing problem, unless there are precedence relationships, in which case it is a combined vehicle routing and scheduling problem)
- B. Number of depots
 - 1. one
 - 2. more than one.
- C. Size of vehicle fleet
 - 1. one
 - 2. more than one

¹The reader is advised that some authors, particularly in the European journals, use the term "vehicle scheduling" when only spatial constraints are involved. This mixing of terms is not so common in the American journals.

- D. Type of fleet available
 - 1. homogeneous case (all vehicles the same)
 - 2. heterogeneous case (not all vehicles the same)
- E. Nature of demands
 - 1. deterministic
 - 2. stochastic
- F. Location of demands
 - 1. at nodes (not necessarily all)
 - 2. on arcs (not necessarily all)
 - 3. mixed
- G. Underlying network
 - 1. undirected
 - 2. directed
 - 3. mixed
- H. Vehicle capacity constraints
 - 1. imposed - all the same
 - 2. imposed - not all the same
 - 3. not imposed
- I. Maximum vehicle route times
 - 1. imposed - all the same
 - 2. imposed - not all the same
 - 3. not imposed
- J. Costs
 - 1. variable or routing costs
 - 2. fixed operating or vehicle acquisition costs
- K. Operations
 - 1. pickups only
 - 2. deliveries only
 - 3. mixed
- L. Objective
 - 1. minimize routing costs involved
 - 2. minimize sum of fixed and variable costs
 - 3. minimize number of vehicles required
- M. Other (problem-dependent) constraints

This taxonomy provides a practical means to classify any routing or scheduling problem. Note that under this system, the classical traveling salesman problem (TSP) would be classified as A3, B1, C1, D1, E1, F1, G1, H3, I3, J1, K1, L1.

The objective function to be minimized can actually be of a hierachal nature. For instance, instead of minimizing the routing costs, a problem formulation might require the minimization of fleet size (L3) and, subject to this minimum fleet size, the minimization of routing costs (L1). Although the minimization of routing costs alone might result in minimum fleet size, there are instances in which the two objectives are not compatible (see, for instance, Eilon et al., 1971, p. 222).

Solution Techniques

The VRP is closely related to the traveling salesman problem (TSP). In the discussion of the TSP, it was pointed out that the TSP is NP-complete, and that problems of moderate size are difficult to solve by exact procedures. This holds more so for the VRP, which is also NP-complete, and which is subject to many more constraints than is the TSP. The classification of solution methodologies which follows in this section is due to Bodin et al. (1983). These procedures include (1) exact methods, (2) cluster first - route second, (3) route first - cluster second, (4) savings and insertion, (5) improvement/exchange, (6) mathematical programming-based, and (7) interactive methods. These different solution methodologies are discussed below.

It should be noted that some procedures for the VRP possess features of more than one of the classifications listed above. In those

cases the procedures are classified according to their most salient features.

Exact Methods. An early formulation of the VRP as an integer program was given by Balinski and Quandt (1964). Their formulation requires only that all customers be served and that vehicle capacities not be exceeded. If m feasible routes are designated beforehand, and if the constant d_{ij} designates whether customer j is included on route i ($1 = \text{yes}$, $0 = \text{no}$), the problem is written as

$$\text{Min } \sum_{i=1}^m x_i c_i \quad (2.3)$$

$$\text{S. T. } \sum_{i=1}^m d_{ij} x_i = 1 \quad \text{for all } j \quad (2.4)$$

$$x_i = 0, 1 \quad \text{for all } i \quad (2.5)$$

where c_i is the cost of delivering to all customers on route i . It can be seen that x_i equals unity if route i is in the solution, and zero otherwise. Balinski and Quandt used Gomory's cutting plane algorithm to solve the 0-1 program.

The formulation given above is not very useful for four major reasons:

1. The number of feasible routes m can be very large for problems of moderate size.
2. Enumerating all possible feasible routes is difficult.
3. To find the cost of supplying customers on the routes, c_i , a traveling salesman problem must be solved for each route.
4. Other constraints cannot be easily appended to the problem.

Foster and Ryan (1976) also used an integer programming formulation

of the VRP. In their method, a feasible set of petal-shaped routes was formulated, and the "over-constrained" problem was solved using linear programming. Cutting planes were used to maintain integrality. The problem was then relaxed to a certain extent to expand the feasible region. Although classified as an "exact" procedure because of the LP approach employed, this method does not guarantee an optimal solution; the time required to relax the constraints to allow consideration of all possible routes is generally prohibitive.

Christofides and Eilon (1969) developed an exact solution procedure based on Little's branch-and-bound tour-building algorithm for the TSP. At each step in the tree, a check is made to ensure that none of the VRP constraints are violated. Also, to prevent unnecessary tree search effort, at each point a feasibility check is made to determine whether sufficient vehicle capacity remains. Bounds are calculated using minimum spanning trees. Using this procedure, VRPs containing up to 12 customers were solved (Eilon et al., 1971).

Christofides, Mingozzi and Toth (1981) present three exact VRP algorithms. Two of the algorithms have bounds calculated from minimum k -degree center trees (k -DCT). A k -DCT is a spanning tree with a given center vertex having exactly k degrees. The third algorithm has bounds calculated from q -routes. A q -route is a least-cost path, from the origin to a node i and back to the origin, for which the total load along the path equals q . In each case, the bounds are calculated by a Lagrangean relaxation ascent method. Computational experience with the algorithms indicate that the bounds calculated from q -routes are superior to those calculated from k -DCTs. VRPs containing up to 25 customers have been solved by this method.

Cluster First - Route Second. This type of procedure solves the VRP in two major steps. The first step assigns customers to individual vehicles by some means, and the second step then sequences the visits among each vehicle's set of customers.

Tyagi (1968) presented an early, very simple version of a cluster first - route second algorithm. His method simply added the nearest neighbor to the last location added, and continued in this manner as long as vehicle capacity constraints allowed. After m routes were formed in this manner, a TSP solution was obtained for each route. According to Golden, Magnanti and Nguyen (1977) the Tyagi algorithm, while computationally attractive, generally yields inferior solutions.

Gillett and Miller (1974) developed a cluster first - route second procedure called the "sweep" algorithm. It derives its name from the manner in which routes are formed by sweeping an imaginary pointer, fixed at the origin, across all the customer locations in a clockwise manner. Customers are added one at a time to a route as long as vehicle capacity is not exceeded. In this way, m routes are formed. A second step in the sweep algorithm considers exchanges between neighboring routes which serve to reduce the overall distance figure. A third step rotates all the locations counterclockwise so that the first location becomes the last, and the whole process is repeated. This continues until each location has served as the first. The best answer from all such rotations is selected. A second procedure called the "backward sweep" algorithm performs the same steps but the rotations are clockwise instead of counterclockwise.

Gillett and Miller found the time to solve a problem to be a function of the number of routes and the number of locations per route.

The time increased linearly with the number of locations if the route sizes remained approximately constant, and the time increased quadratically with the number of locations per route if the total number of locations remained approximately constant. Problems up to 250 locations were solved using this method.

Other cluster first -route second approaches have been developed by Gillett and Johnson (1976), Chapleau et al. (1981), and Evans and Norback (1984).

Route First - Cluster Second. These procedures work in the reverse order to those discussed above. First, an overall TSP is solved over all the locations, then this overall tour is partitioned into feasible subsets. Several authors have presented versions of the route first - cluster second methodology. Among them are Bodin and Berman (1979), Beasley (1983), Ball et al. (1983) and Golden et al. (1984). The method presented below is due to Beasley (1983):

Let d_{ij} represent the distance between any two locations. Using this distance matrix, a grand tour is formed through all locations, excluding the depot. The locations are renumbered as they appear on the grand tour (depot = 0). Now, define c_{ij} to be the cost of supplying customers $(i + 1, i + 2, \dots, j)$ in any order. This cost is the amount it takes to add a new route containing customers $(i + 1, i + 2, \dots, j)$, given that customer i has been served on a route.

Using the cost matrix thus formed, a network problem is formulated and the shortest path between points is found. The number of vehicles required is equal to the number of arcs containing in the shortest path.

Because the depot is not included in Beasley's formulation, the costs c_{ij} are found by solving a small TSP for each set of locations

(0, i + 1, i + 2, . . . , j). His entire procedure is summarized below:

- Step 1. Generate grand tour and use a 2-opt procedure to improve it until further improvements cannot be made using a 2-exchange.
- Step 2. Calculate matrix (c_{ij}) using a 2-opt procedure to find TSP solution among customers (i+1, i+2, . . . , j). Add vehicle cost to each c_{ij} .
- Step 3. Use Floyd's algorithm to calculate the least-cost paths through the directed graph formed by (c_{ij}), thereby partitioning the grand tour.
- Step 4. Use a 3-opt procedure to optimize each individual partition.

Savings/Insertion. The savings method of Clarke and Wright (1964) was covered previously for the TSP. To be applicable to the VRP, the method is altered in only one respect: Each time a link is to be formed by joining two routes together, a feasibility check is made to determine whether vehicle capacity or route duration constraints are met. If so, the routes are joined; if not, the savings link is ignored and the search through the savings list is continued.

Many modifications have been made to the original (Clarke and Wright) version of the savings method. For instance, one of the shortcomings of the original method is that links between locations, once formed, are permanent throughout the duration of the procedure. Tillman and Cochran (1968) proposed a method which overcomes this to some extent. In their method, two routes are not joined permanently until a check is made to determine if the savings might be greater if the connection is not made. A comparison is made among the overall savings resulting from

initial connection of the routes showing the highest savings, second-highest, third-highest, etc. Tillman and Cochran reported computational results for only one problem; their algorithm achieved a 6.2 percent reduction in distance over the original Clarke and Wright savings algorithm. A similar approach, involving the "suppression" of savings links already in a given solution and the re-solving of the problem without that link, was reported by Holmes and Parker (1976). Beltrami and Bodin (1974) found that improvements could sometimes be made by simply perturbing the ordering within the savings list, by artificially increasing the distance from the depot to one or more locations, then re-solving the problem subject to the original constraints.

Gaskell (1967) observed that the Clarke and Wright savings method tends to produce peripheral routes which sometimes overlap. To overcome this, the following two savings measures were proposed:

$$1. \lambda_{ij} = S_{ij}(\bar{d} + d_{0i} - d_{0j} - d_{ij}) \quad (2.6)$$

$$2. \pi_{ij} = S_{ij} - d_{ij} \quad (2.7)$$

where S_{ij} = Clarke and Wright savings,

d_{ij} = distance between locations i and j,

and \bar{d} = average distance from depot to all locations.

Both of these measures are modifications of the Clarke and Wright savings measure, and tend to give more emphasis to radially-aligned routes.

The savings algorithm of Clarke and Wright is sometimes known as a "concurrent" or "multiple" version of the method, due to the fact that multiple routes are formed at the same time. While having the advantage that evolving routes compete with one another, the concurrent version

has the disadvantage of requiring the entire savings file to be maintained in storage at one time. Yellow (1970), Webb (1971), and Mole and Jameson (1976) introduced "sequential" versions of the savings method which do not require the large savings file. Sequential methods are so named because a route, once begun, is continued as long as feasibility conditions are met. In sequential methods, no competition between routes is present.

An insertion procedure developed by Williams (1982) is known as "proximity priority searching." Beginning with the location most distant from the depot, a new link is formed and the closest two feasible nodes are "pseudo-assigned" (i.e., temporarily assigned) to the link. From that point, the method proceeds as follows:

Step 1. Consider the location next most distant from the depot. If this location has been pseudo-assigned to a link end, make the connection permanent, and find the closest feasible location for pseudo-assignment to the new link end. However, if this (next most distant) location has not been pseudo-assigned to an existing link, add the location to the link list to form the beginning of a new link, and pseudo-assign the closest two feasible nodes.

Step 2. If the closest location is the end of another link, the two links are joined together if a feasible route results.

Step 3. Once a link meets restrictions on load and distance, it is considered complete and is not used for further node assignments.

Step 4. If all nodes have not been assigned, go to Step 1.

This proximity priority searching procedure was tested against six other heuristics over eight problems, and performed as well as any of the others, obtaining the optimal solution in five of the eight problems. A significant fact is that a microcomputer (Cromenco Z2H) was used in Williams' research. The longest computing time was two minutes for a 50-node problem.

Savings and insertion procedures have been extended to the multiple-depot problem. If each city in a VRP is first assigned to the nearest of m depots, and then the Clarke and Wright savings is applied directly to those cities, the indicated savings will not be correct. This is because the savings for linking two cities which are close to one terminal and a greater distance from a second terminal would indicate that the cities should be linked to the farthest terminal, which is incorrect. Tillman and Cain (1972) used a "modified distance" and "modified savings" to overcome this problem. The modified distance is calculated by:

$$\bar{d}_i^k = \min_m d_i^m - (d_i^k - \min_m d_i^m) \quad (2.8)$$

and the modified savings is calculated by:

$$\bar{s}_{ij}^k = \bar{d}_i^k + \bar{d}_j^k - d_{ij} \quad (2.9)$$

where d_{ij} = distance between cities i and j ,

and d_i^k = distance from depot k to city i .

In addition to the savings measure, Tillman and Cain introduced a penalty factor for not creating a particular link, and used a weighted

combination of the savings and penalty measures in determining which locations to link together. Golden et al. (1977) used the Tillman and Cain approach as the basis for a multiple-terminal algorithm which requires fewer computations at each step and which uses fewer storage locations for the savings matrix.

Improvement/Exchange. These methods begin with an initial solution to the VRP and make improvements to that solution by exchanging the relative positions of customers on a route or between routes.

Christofides and Eilon (1969) applied the 3-opt method of Lin (1965), which has been covered previously in this chapter in discussing the TSP. The initial tour of Christofides and Eilon included the same number of (artificial) depots as there were vehicles available. Feasibility checks for vehicle capacity and tour length were made at each exchange. The method produced solutions superior to the savings approach, although at a premium in computational effort. Russell (1977) used a similar approach, except he employed the k-opt method of Lin and Kernighan (1973).

Improvement and exchange heuristics have been applied to the multiple depot problem as well. Newton and Thomas (1974) employed an algorithm similar to Lin's 3-opt method in solving school bus scheduling problems. However, their algorithm was applicable to asymmetric VRPs, so the branch exchanges were limited to those which would not alter the direction of travel in the unchanged portion of the route. Wren and Holliday (1972) applied seven different improvement routines to a set of routes which were initially formed by an insertion procedure. These improvement routines allowed the movement of customers within a route or between routes, as well as the consolidation and recombining of two routes into one.

Mathematical-Programming Based. These procedures, instead of employing "rule of thumb" heuristics, employ heuristics which are based on a mathematical programming formulation of the VRP. A good example of this approach is given by Fisher and Jaikumar (1981). If

K = number of vehicles

n = number of customers

b_k = capacity of vehicle k

a_i = demand of customer i

c_{ij} = travel cost (time, distance, or dollars) from i to j

$y_{ik} = \begin{cases} 1, & \text{if customer } i \text{ is served by vehicle } k \\ 0, & \text{otherwise} \end{cases}$

$x_{ijk} = \begin{cases} 1, & \text{if vehicle } k \text{ travels from customer } i \text{ to } j \\ 0, & \text{otherwise} \end{cases}$

then the VRP is given by

$$\text{Min } \sum_{ijk} c_{ij} x_{ijk} \quad (2.10)$$

$$\text{S. T. } \sum_i a_i y_{ik} \leq b_k \quad k = 1, 2, 3, \dots, K \quad (2.11)$$

$$\sum_k y_{ik} = \begin{cases} K, & \text{if } i = 0 \\ 1, & \text{if } i = 1, 2, 3, \dots, n \end{cases} \quad (2.12)$$

$$y_{ik} = 0, 1 \quad i = 0, 1, 2, 3, \dots, n \quad (2.13)$$

$$k = 1, 2, 3, \dots, K$$

$$\sum_i x_{ijk} = y_{jk} \quad j = 0, 1, 2, 3, \dots, n \quad (2.14)$$

$$\sum_j x_{ijk} = y_{ik} \quad i = 0, 1, 2, 3, \dots, n \quad (2.15)$$

$$\sum_{i,j \in S \times S} x_{ijk} \leq |S| - 1 \quad S \subseteq \{1, 2, 3, \dots, n\} \quad (2.16)$$

$$2 \leq |S| \leq n-1$$

$$x_{ijk} = 0, 1 \quad i = 0, 1, 2, 3, \dots, n \quad (2.17)$$

$$j = 0, 1, 2, 3, \dots, n$$

Here, it can be seen that constraints (2.11) through (2.13) apply to a generalized assignment problem, and constraints (2.14) through (2.16) apply to a traveling salesman problem over the customers assigned (by the generalized assignment problem) to a given vehicle k . In reformulating the VRP, Fisher and Jaikumar replace the objective above with the function

$$\text{Min } \sum_k f(y_k) \quad (2.18)$$

where $f(y_k)$ is the cost of an optimal TSP tour of the customers $\{i \mid y_{ik} = 1\}$. This function is defined mathematically as

$$f(y_k) = \min \sum_{ijk} c_{ij} x_{ijk} \quad (2.19)$$

subject to the TSP constraints given above. Since $f(y_k)$ is a very complicated function, a linear approximation is used, instead:

$$\sum_{i=1}^n d_{ik} y_{ik}. \quad \text{Then the objective becomes}$$

$$\text{Min } \sum_k \sum_{i=1}^n d_{ik} y_{ik}. \quad (2.20)$$

Solving the generalized assignment problem determines a set of customers for each vehicle. A tour is then constructed through each set using any TSP procedure (Fisher and Jaikumar use Miliotis' exact method, but a heuristic TSP method should also work well).

The values of d_{ik} are determined by first assigning a "seed" customer i_k to each vehicle 1, 2, 3, . . . , K. Then d_{ik} is the cost of inserting customer i into the route from the depot to the seed customer i_k .

$$d_{ik} = \min_k [c_{0i} + c_{ii} + c_{i0} , c_{0i} + c_{ii} + c_{i0}] - [c_{0i} + c_{i0}] \quad (2.21)$$

Cheshire et al. (1982) use a dual heuristic to create a set of routes. In their method, constraints pertaining to vehicle load limits and route time limits are allowed to be violated in the construction of the routes, but at a penalty $p \cdot k$, where p is a constant expressed in distance units and k is a Lagrangean multiplier. In calculating the cost of inserting a customer into a route, the extra distance required by the insertion and the penalty $p \cdot k$ are added together. At any step, the lowest-cost customer is inserted into the appropriate position. After all customers have been included in the routes, any constraint violations are reduced by iteratively increasing the value(s) of the Lagrangean multiplier(s). In this manner, the procedure maintains local optimality while approaching feasibility.

Stewart and Golden (1984) use a similar approach in their LR3OPT algorithm. Their formulation for the VRP with m vehicles, each having capacity Q , is:

$$\text{Min } \sum_k \sum_{ij} c_{ij} x_{ijk} + \sum_k \lambda_k (\sum_{ij} \mu_i x_{ijk} - Q) \quad (2.22)$$

$$\text{S. T. } \sum_{ij} \mu_i x_{ijk} - \sum_{ij} \mu_i x_{ijl} \geq 0, \quad l = k+1 \quad (2.23)$$

$$k = 1, 2, 3, \dots, m-1$$

$$\lambda_k > 0, \quad k = 1, 2, 3, \dots, m \quad (2.24)$$

where μ_i is the demand at location i . The second half of the objective function is the result of moving capacity constraints, via Lagrangean relaxation, out of the constraint set. The remaining constraint set is simply a method of numbering the routes so that the route which exceeds

vehicle capacity by the greatest amount is numbered 1, the route which exceeds vehicle capacity by the next greater amount is numbered 2, etc. The λ s are Lagrangean multipliers. In practice, only λ_1 is used in the objective function, which is written as

$$\text{Min } \sum_k \sum_{ij} c_{ijk} x_{ijk} + \lambda_1 \sum_{k \in K^+} (\mu_i x_{ijk} - Q) \quad (2.25)$$

$$\text{where } K^+ = \{k \mid \sum_{ij} \mu_i x_{ijk} > Q\}. \quad (2.26)$$

The algorithm uses a normal 3-opt exchange procedure to obtain the minimum at each value of λ_1 . The values λ_1 are increased geometrically until a feasible solution is found. Published results indicate that the LR3OPT algorithm performs similarly to the dual heuristic of Cheshire et al. (1982).

Interactive. These methods employ a man-machine interface to solve the VRP. Each participant (man and machine) provides input to that part of the problem-solving process for which that participant is best suited. For instance, suitable pairwise changes in a Euclidean routing problem are sometimes obvious to a decision maker when the routes are displayed to him. After the decision maker indicates the pair(s) to be exchanged, the computer can quickly and efficiently calculate the results of the exchange.

There are two types of interaction which can exist between the human decision maker and the computer. First is the case in which the decision maker actually provides part of the solution by making absolute directives to the computer. Moving a customer from one route to another is an example of this type of interaction. "Locking out" a route from the problem is another such example. The second type of interaction is

one in which the decision maker guides the progress of problem solution by progressively articulating his preferences to the computer as the solution proceeds. The interactive sequential goal programming procedure of Park (1984) in the solution of multicriteria VRPs contains examples of this type of interaction.

Krolak, Felts and Nelson (1972) developed a two-phase interactive procedure. In the first phase, customers are clustered based on their closeness to each other, without regard to demands, using a standard transportation algorithm. Then these clusters are repeatedly joined pair-wise, based upon the distances between their centers of gravity, until the resulting number of clusters is equal to the prespecified number of routes. The transportation algorithm is again applied to the clusters, resulting in an assignment of vehicles to routes. A TSP routine is applied to each route, then customers are moved from one route to the next until capacity constraints are met. The TSP algorithm is reapplied to the feasible routes, and then simple swapping algorithms are applied in alternation with the feasibility and TSP algorithms until time expires or a specified number of iterations has been performed.

The second phase is the interactive part of the procedure. The decision maker is able to request displays of the current routes, routes in previous solutions, loads and coordinates of customers and vehicles, overloaded vehicles, and current cost. Based upon this information, he may move single customers or strings of customers from one route to another, or exchange a pair of customers in a given route. He may also apply regional heuristics; that is, one or more routes may be isolated for further refinement by one of the VRP heuristics included in the package. Krolak, Felts and Nelson reported that, while the displays

provided by Euclidean problems are most comprehensible to the decision maker, other distance metrics can be employed without significantly affecting the ability of the decision maker to recognize possible improvements.

Lawrence (1981) experimented with an interactive procedure which allowed the decision maker to choose from among three different VRP algorithms and three different TSP algorithms in forming and improving a set of routes. A refinement phase then allowed the user to make minor modifications to those routes. Computer graphics were used extensively throughout the interactive package developed by Lawrence.

Scion Consultancy in Great Britain has marketed an interactive VRP package called VANPLAN. Stacey (1983) has reported on a case study involving its implementation. He concluded the following advantages to the interactive method:

1. Non-quantifiable data, such as customer likes and dislikes, can sometimes be traded off against more efficient routes.
2. The system can respond easily to changing parameters.
3. Driver acceptability of the routes is not a concern since an experienced load planner is involved in the route formation.
4. Staff training is less than it would be if algorithmically-planned routes were used.
5. Dependency on a highly skilled staff is reduced.

Stacey also noted the following advantages:

1. Some monitoring of the operators is necessary at first.
2. Management usually has a fear that the best solution will be missed.

Waters (1984) reports on another study involving an interactive routing package. Ten problems were solved and compared, with results obtained by Clarke and Wright (1964), Wren and Holliday (1972), and Foster and Ryan (1976). In general, the total distance obtained by the interactive procedure compared very favorably with that obtained by the other three. Total "hands on" solution times ranged from five minutes for a 6-customer problem to forty-five minutes for a 100-customer problem.

An interactive model used to solve multicriteria VRPs is reported by Park (1984). The VRP under consideration has the following three objectives:

1. Minimize total distance traveled.
2. Minimize deterioration of goods during transport.
3. Maximize fulfillment of priority deliveries and inter-customer precedence requirements.

Park's model uses an "iterative goal programming heuristic" approach. The decision maker can interact with the computer by progressively articulating his preferences through changes in the priority structure or goal levels, or by directly moving a customer from one route to another. No actual "hands on" solution times are reported for this model.

Workload Balancing in VRPs

Before discussing procedures to balance crew workloads, a short discussion regarding VRP objectives is in order. The original works of Dantzig and Ramser (1958) and Clarke and Wright (1964) regarded the minimization of distance as the VRP's single objective, and many others have followed with the same objective. Other authors (e.g., Gaskell,

1967; Christofides and Eilon, 1969; Foster and Ryan, 1976; and Cheshire et al., 1982) stated their objectives to be hierachal; i.e. first, the minimization of the number of routes and second, the minimization of both the number of routes and the distance traveled. The fact that some of these formulations involve conflicting objectives received little notice in earlier works. However, Wren and Holliday (1972) observed that

The cost factors generally considered are the number of trucks, and the total distance traveled. There may be a conflict of objectives here; such conflict has not been resolved in the past. Experience with the current program indicates that no conflict in fact exists (p. 333).

Although Wren and Holliday discounted the conflict between the two objectives, others have considered such conflict. Eilon, Watson-Gandy and Christofides (1971, p. 223) devised an example of such vehicle/distance conflicts. In their example, the total distance of a problem was decreased by adding an extra vehicle to the solution. A similar example occurred in a comparison of five VRP heuristics by Christofides, Mingozi and Toth (1979, p. 335). This example, which went without comment by the authors, showed a reduction of one vehicle for the Mole and Jameson algorithm over the SWEEP algorithm of Gillett and Miller, but at a distance penalty of slightly greater than four percent.

A question naturally arises regarding the use of single-objective algorithms to solve multiple-objective problems. Just how is the final solution chosen in the face of (non-hierachal) conflicting objectives? Little insight is provided by the authors of these earlier sources. It can only be assumed that, if different trials of a method produced more than one efficient solution to a given problem, the authors were willing to employ tradeoffs as necessary to choose between them.

Although not part of a particular problem formulation, some authors

have recognized that objectives other than distance and/or number of vehicles are important in real-world applications. Golden, Magnanti and Nguyen (1977, p. 125) refer to "unstated goals and/or constraints", and suggest that several solutions be generated by varying a route shape parameter, from which the decision maker can supposedly choose the best solution. Lawrence (1981, p. 102), in detailing the advantages of the interactive man-machine approach to solving the VRP, observed that "unstated objectives and constraints which are difficult to express need not be explicitly stated to the machine." The human decision maker can presumably interact with the machine in such a way that the "unstated objectives" are optimized while the "unstated constraints" are met.

Several authors have alluded to the problem of providing some measure of equity among the tasks assigned to the driver force. For example, Kirby and McDonald (1973) criticized the routes formed by the savings method:

Routes are often produced which would be quite unacceptable to any transport manager; for example, those which cross themselves, which cross other routes at more than one point, and solutions which include too many short routes. In our opinion, these difficulties call into question the normally accepted criterion of 'optimality' and suggest that there is a place for a subjective factor in assessing optimality (p. 305).

Mole (1979) states that

In practice, a dispatcher will also attempt to reconcile the immediate task of providing efficient schedules with a broader view of the business, such as the need to retain equity as between the tasks assigned to several drivers . . . (p. 246).

And finally Waters (1984), in arguing the case for interactive procedures, observes:

Traditionally, computerized vehicle scheduling has concentrated on minimizing total distance travelled and has largely ignored other objectives, such as minimization of fleet size, variable costs, elapsed time or total travelling time. Other factors for

consideration might include an equitable distribution of workload amongst drivers and vehicles . . . (p. 821).

Evans and Norback (1984) have developed an algorithm to solve the "time sensitive vehicle routing problem". One of the goals of their method is to obtain routes which satisfy certain restrictions on total route time and driver workloads. For instance, a particular application might require that at least half of the routes contain between eight and ten hours of driver work time. Through the use of a "time density function", customers are first clustered and then routes are formed through the clusters. The time density function is such that heavy insertion penalties are associated with customers near the depot, so these customers are usually assigned to the last route formed. If this last route contains fewer than a pre-specified number of customers, the method attempts to redistribute these customers among the other routes. All of these customers, being near the depot, are also relatively close to each of the other routes. This makes the chances good that these redistributed customers can be used to balance the driver workloads while adding minimal distance to the routes.

Dileepan (1984), in solving the "delivery planning problem", observes that the long-term cost of delivery to a fixed set of customers whose demands vary from period to period is made up to two components: (1) the routing costs within a given period and (2) the cost of changing the routes between periods to ensure feasibility. The number of such route changes can be affected by two factors: (1) the variability of demand from period to period, and (2) the degree to which the routes are balanced. In Dileepan's work, a measure of imbalance is given by the sum of squared deviations in workload (total route time). The objective function to be minimized is a convex combination of the total delivery

times and the workload imbalance. To solve the problem

$$\text{Min } \lambda \cdot \text{total delivery time} + (1 - \lambda) \cdot \text{workload imbalance},$$

eleven different values of λ (0, .1, .2, . . . , 1.0) are selected, and each such problem is solved heuristically using a branch-exchange procedure. In this way, eleven different efficient solution points are generated. The final selection of the best solution is obtained via a computer simulation over a complete planning period which indicates the total impact (delivery cost plus number of changes) over the planning period. Of course, there is no guarantee that all efficient points have been generated by the procedure.

The generation of each efficient point for a 100-customer problem using Dileepan's method took one minute of cpu time on a NAS9000 computer. By extrapolation, the generation of all eleven points of the efficient set should require approximately eleven minutes cpu time. If the number of objectives were increased to three (instead of the two in Dileepan's algorithm), and if the convex multiplier λ is still increased in increments of 0.1 as before, then a minimum of one hour cpu time could be expected to generate the 66 efficient points of the problem.

Husban (1985) employs a "route first -cluster second" arc-routing heuristic in solving the Balanced Tractor-Trailer Routing Problem (BTTRP). In this type of problem, demand is expressed in "move orders" between pickup and delivery points. Because only full trailer loads are transported between points, vehicle capacity is not a concern. To balance the distance traveled by a trailer, Husban uses a minimax criterion, minimization of the longest route. This criterion does not include total time or distance, so a combination of minimum time or

distance and the balancing criterion is used as an objective function. The weighting factor used in this combined function is assumed to be known.

Benton (1986) uses a sequential version of the Clarke and Wright savings algorithm to develop initial route assignments, then optimizes each route thus formed by use of Little's tour-building algorithm. Benton claims that this technique results in route sets which are balanced in terms of driving times; however, his comparisons with the original Clarke and Wright algorithm over sixty distance matrices do not include values of an imbalance measure. Mean travel time is used, instead.

Summary

This chapter has reviewed the various types of procedures which have been used to solve the vehicle routing problem, beginning with a review of the closely related traveling salesman problem. Both exact and heuristic techniques were covered for each of these problems.

Exact procedures for the TSP include dynamic programming, linear and integer programming, and branch-and-bound approaches. Heuristic procedures include tour-building, tour-improvement, and composite methods. VRP exact procedures include integer programming and branch-and-bound methods, including the use of Lagrangean relaxation techniques to compute tight lower bounds. VRP heuristic procedures include cluster first - route second, route first - cluster second, savings/insertion, improvement/exchange, mathematical programming-based, and interactive methods.

Earlier approaches to solving the VRP used single-objective

procedures, even if the existence of conflicting objectives was recognized. These conflicting objectives were sometimes placed in a hierachal ranking, and multiple solutions could be compared using this hierarchy, although the single-objective algorithms were not designed to optimize more than one objective. For instance, if the hierarchy of objectives was minimization of the number of vehicles first and minimization of distance traveled second, then multiple solutions from the algorithm would be evaluated by choosing those having the minimum number of vehicles and, within that set, those with the minimum distance traveled. The algorithm, however, "aimed" at only a single (minimum-distance) objective. A notable exception is the algorithm of Park (1984), which uses a heuristic goal programming approach to consider more than a single objective.

As can be seen by the comparison of VRP algorithms in Table 2.1, workload balancing in VRPs has received little attention in the literature. Of the two workload elements in the WBVRP, only the driving time or distance element has received balancing treatment. To do this, two approaches have been taken: (1) incorporate balancing goals in the form of constraints and solve the VRP using a distance-minimizing algorithm, and (2) form several convex combinations of the delivery-time and imbalance objective functions and minimize each resulting function, then select the preferred solution out of the efficient set thus generated. Neither of these approaches provides for an interactive "learning" environment for the decision maker. In addition, the second approach can lead to the solving of a large number of problems, requiring an excessive amount of computer time.

TABLE 2.1

VRP ALGORITHMS

Solution Strategy	Algorithm	Year	Objectives	Route Balancing?	Notes
			Single Mult.		
Exact	Balinski & Quandt	1964	x	No	
"	Christofides & Eilon	1969	x	No	
"	Foster & Ryan	1976	x	No	
"	Christofides et al.	1981	x	No	
Cluster First, Route Second	Tyagi	1968	x	No	
	Gillett & Miller	1974	x	No	
"	Gillett & Johnson	1976	x	No	
"	Evans & Norback	1984	x	Yes	1
Route First, Cluster Second	Bodin & Berman	1979	x	No	
	Beasley	1983	x	No	
"	Ball et al.	1983	x	No	
"	Golden et al.	1984	x	No	
"	Husban	1985	x	Yes	2
Savings/Insertion	Dantzig & Ramser	1959	x	No	
"	Clarke & Wright	1964	x	No	
"	Gaskell	1967	x	No	
"	Tillman & Cochran	1968	x	No	
"	Yellow	1970	x	No	
"	Webb	1971	x	No	
"	Tillman & Cain	1972	x	No	
"	Beltrami & Bodin	1974	x	No	
"	Holmes & Parker	1976	x	No	
"	Mole & Jameson	1976	x	No	
"	Golden et al.	1977	x	No	
"	Williams	1982	x	No	
"	Benton	1986	x	Yes	3
Improvement/Exch.	Christofides & Eilon	1969	x	No	
"	Wren & Holliday	1972	x	No	
"	Newton & Thomas	1974	x	No	
"	Russell	1977	x	No	
"	Dileepan	1984	x	Yes	4
Math Prog. Based	Fisher & Jaikumar	1981	x	No	
"	Cheshire et al.	1982	x	No	
"	Stewart & Golden	1984	x	No	
Interactive	Krolak et al.	1972	x	No	
"	Lawrence	1981	x	No	
"	Stacey	1983	x	No	
"	Waters	1984	x	No	5
"	Park	1984	x	No	

¹Driving times balanced through constraints.²Balanced Tractor-Trailor Routing Problem.³Driving times balanced by sequential savings.⁴Convex combination of objectives. Near-efficient set generated.⁵Multiple objectives not specifically stated, but supposedly known by scheduler.

CHAPTER III

GENERAL MODEL STRUCTURE

In this chapter, a general approach is developed to solve the WBVRP. The problem is viewed in the context of multicriteria optimization, and a model structure suitable for its solution in this context is presented. The actual heuristics used to solve the problem are not presented here, but are covered in the next chapter.

Assumptions

In order to begin model development, certain assumptions are necessary. The assumptions which have been adopted for this research are:

1. The vehicle fleet is homogeneous (equal capacity), or the route planner is willing to accept the least capacity in the fleet as a constraint for all vehicles in the fleet.
2. Demand at each customer location is known and deterministic.
3. The distance matrix is symmetric, but not necessarily Euclidean. If non-Euclidean distances are to be used, it is assumed that the distance between each pair of customer locations has been determined by a distance-minimizing procedure.
4. Physical route limitations such as one-way streets, traffic

- congestion, road conditions, detours, etc. are ignored.
5. Individual routes are constrained by vehicle capacity, and may or may not be constrained by maximum distance, depending on the problem characteristics.
 6. No explicit utility function is assumed. Instead, an interactive learning environment for the route planner is desired.
 7. Satisficing solutions are acceptable. This implies the acceptability of non-optimal solutions.
 8. For workload balancing purposes, a crew is assigned to only one route. This precludes the grouping together of two or more short routes in order to balance the driving distances.
 9. User interaction with the model may be of two types:
 - a. Preemptive
 - b. Preference articulation

Mathematical Representation of the WBVRP

The WBVRP is now formulated as an integer program having three objective functions. Assume there are K vehicles, each having a given capacity. Define the problem as

$$\text{Min } f_1 = \sum_k \sum_{i,j} c_{ij} x_{ijk} \quad (3.1)$$

$$\& \quad f_2 = \sum_{i,j} c_{ij} x_{ije} - \sum_{i,j} c_{ij} x_{ijf} \quad (3.2)$$

$$\& \quad f_3 = \sum_{i,j} d_j x_{ijg} - \sum_{i,j} d_j x_{ijh} \quad (3.3)$$

$$\text{S. T. } x_{ijk} \in S_K \quad \forall ijk \quad (3.4)$$

$$x_{ijk} = 0, 1 \quad \forall ijk \quad (3.5)$$

$$\sum_{i,j} d_j x_{ijk} \leq C \quad k = 1, 2, 3, \dots, K \quad (3.6)$$

$$\sum_{i,j} c_{ij} x_{ijk} \leq D \quad k = 1, 2, 3, \dots, K \quad (3.7)$$

$$\sum_{i,j} d_j x_{ijk} - \sum_{i,j} d_j x_{ijl} \geq 0 \quad \text{for } l = k+1 \quad (3.8)$$

& k = 1, 2, 3, \dots, K-1

$$\sum_{i,j} c_{ij} x_{ijm} - \sum_{i,j} c_{ij} x_{ijn} \geq 0 \quad \text{for } n = m+1 \quad (3.9)$$

& m = 1, 2, 3, \dots, K-1

where f_1 = total distance

f_2 = route length deviation

f_3 = route load deviation

c_{ijk} = distance traveled between i and j by vehicle k

x_{ijk} = binary variable indicating whether i and j are served by vehicle k (yes = 1, no = 0)

d_j = demand at location j

C = vehicle capacity

D = route distance limit

S_K = set of feasible solutions to the K-TSP

e = vehicle serving route with greatest distance

f = vehicle serving route with least distance

g = vehicle serving route with greatest demand

h = vehicle serving route with least demand

Equations (3.1) through (3.9) define a vector-maximum version of the VRP having K vehicles. Objective function (3.1) calls for the minimization of total distance, (3.2) calls for the minimization of route length deviation, and (3.3) calls for the minimization of route load deviation. Constraint set (3.6) defines the load limit of each vehicle, and

constraints (3.7) define the maximum length of each route. Equations (3.8) and (3.9) serve as a means of numbering the routes from the longest to shortest and from most heavily loaded to least heavily loaded, respectively (much in the same manner as the model of Stewart and Golden, 1984).

Multiple Criteria Optimization

The WBVRP, because it contains three different objectives (minimization of total distance, route load deviation, and route length deviation), is a multicriteria optimization problem. The purpose of this section is to provide a brief overview of the area of optimization using multiple criteria.

Terminology

Several terms are used in describing multiple criteria optimization. The following definitions are given by Zeleny (1982):

1. Attributes - descriptors of objective reality. This term refers to traits which can be measured, either objectively or subjectively. In the WBVRP, the attributes are total distance and route-length deviation measures (miles, kilometers, etc.) and measures of route-load deviation (pounds, gallons, number of passengers, number of customers, etc.).
2. Objectives - directions of improvement or preference along individual attributes or complexes of attributes. The three objectives in the WBVRP are in the direction of minimization.
3. Goals - particular target levels of achievement which can be defined in terms of both attributes and objectives. A multiple

criteria optimization problem may or may not include goal values, depending upon the solution methodology employed.

4. Criteria - all the attributes, objectives, or goals which have been judged relevant in a given decision situation by a particular decision maker.

Methods

All multiple criteria optimization problems must contain a set of quantifiable objectives, a constraint set, and a means of obtaining tradeoff information (explicit or implicit) between the objectives.

Hwang and Masud (1979) provide a taxonomy of multiple criteria decision making based upon the stage in the decision process in which the trade-offs are obtained via the decision maker's articulation of preference information. The following discussion follows their taxonomy.

No Preference Information Given. In this category, the decision maker provides no tradeoff information at all, accepting instead the solution provided by the method without qualification. The principal method in this category is the method of global criterion (Boychuk and Ovchinnikov, 1973) in which a global criterion, such as the deviation sum of squares from the feasible ideal points, is minimized. The major disadvantage of this type of procedure is that solutions which are totally unacceptable to the decision maker may be generated.

'A Priori' Preference Articulation. Methods in this category require the decision maker to supply preference information prior to solving the problem. Utility function methods (Keeny, 1972), (Fishburn, 1974), and (Farquhar, 1977) convert the problem to

$$\text{Max } U(f_1, f_2, \dots, f_k) = U(\underline{f}) \quad (3.10)$$

$$\text{S.T. } g_j(x) \leq 0 \quad j = 1, 2, 3, \dots, m \quad (3.11)$$

where $U(\underline{f})$ is the overall utility function of the K multiple objectives.

The problem is then solved using any suitable single-objective optimization technique. The difficulty with utility function methods is that the determination of $U(\underline{f})$, even for small problems, is not an easy task.

Bounded objective methods (Hwang and Masud, 1979) require the decision maker to supply a minimum acceptable level of achievement for each objective function. The problem is converted to

$$\text{Max } f_r(x) \quad (3.12)$$

$$\text{S. T. } g_i(x) \leq 0 \quad i = 1, 2, 3, \dots, m \quad (3.13)$$

$$f_j(x) \geq L_j \quad j = 1, 2, 3, \dots, K \quad (3.14)$$

where L_j is the minimum acceptable achievement level of the j th objective function. Since the decision maker must supply values of L_j in an information void, the method is difficult to apply. In such information voids, the creation of inconsistent constraint sets is possible. Also, it is not always apparent which objective function should be used for $f_r(x)$. Bounded objective methods are rarely used alone, although they may be used as part of other methods (e.g., Benson, 1975).

Goal programming methods have been developed by Charnes and Cooper (1961), Lee (1972), Ignizio (1976), and others. These methods require the decision maker to set goals beforehand for each of the objectives in the problem. The optimal solution to the problem is one in which the deviations from these goals are minimized. The lexicographic version of the method requires the decision maker to also supply an ordinal ranking

of the objectives. The problem in K objectives is expressed as

$$\text{Min } P_1 h_1(\underline{d}^-, \underline{d}^+), P_2 h_2(\underline{d}^-, \underline{d}^+), \dots, P_K h_K(\underline{d}^-, \underline{d}^+) \quad (3.15)$$

$$\text{S. T. } g_j(x) \leq 0 \quad j = 1, 2, 3, \dots, m \quad (3.16)$$

$$f_i(x) + \underline{d}_i^- - \underline{d}_i^+ = b_i \quad i = 1, 2, 3, \dots, K \quad (3.17)$$

$$\underline{d}_i^-, \underline{d}_i^+ \geq 0 \quad \forall i \quad (3.18)$$

$$\underline{d}_i^- \cdot \underline{d}_i^+ = 0 \quad \forall i \quad (3.19)$$

The P_i 's are preemptive weights; i.e., $P_i \gg P_{i+1}$. This implies that

there is no value w which will make $w \cdot P_{i+1} > P_i$. The terms \underline{d}_i^+ and \underline{d}_i^- are positive and negative deviations from the i th goal, respectively.

The $h_i(\underline{d}^-, \underline{d}^+)$ are linear functions of the deviations and are referred to as achievement functions.

Progressive Preference Articulation. These methods, usually called 'interactive' methods, do not require the decision maker to express any preferences beforehand. Instead, the decision maker must provide only local tradeoff information as the methods proceed from one solution to the next. As the methods progress, the decision maker also learns more about the problem being solved. Hwang and Masud (1979) list the following advantages to this type of procedure:

1. There is no need for preference information beforehand.
2. A learning process is involved.
3. Only local preference information is required.
4. There is a greater chance of implementation, since the decision maker is more actively involved in obtaining the solution.

5. There are less restrictive assumptions compared with 'a priori' preference articulation methods.

Tradeoffs by the decision maker can be either explicit or implicit. Those methods involving explicit tradeoffs require the decision maker to choose between specific achievement levels of the objectives. Included in this class are the method of Geoffrion et al. (1972), Dyer's Interactive Goal Programming (1972), the Surrogate Worth Tradeoff Method (Haimes, Hall, and Freedman, 1975), the Method of Satisfactory Goals (Benson, 1975), and the method of Zonts and Wallenius (1976).

Those methods involving implicit tradeoff information do not require the decision maker to choose between specific achievement levels of the objectives. Instead, only acceptable achievement levels must be indicated. There are two advantages to this. First, the decision maker is usually more confident in expressing those acceptable achievement levels. Second, the decision maker does not have to be concerned with the range of validity of tradeoffs, which is usually quite narrow for the methods requiring explicit tradeoffs. Methods in this class include the Step-method (STEM) of Benayoun et al. (1971), Zeleny's Displaced Ideal (1982), GPSTEM (Fichefet, 1976), and Steur's Interactive MOLP (1977).

'A Posteriori' Preference Articulation. In these methods, a subset of nondominated solutions is generated. Then from this subset, the decision maker must choose the preferred solution. The major disadvantage of these methods is that the set of solutions from which the decision maker must choose is usually very large. For this reason, they are usually incorporated into interactive methods instead of being used alone. Methods in this class include the Parametric Method (Gal and

Nedoma, 1972), the ϵ -Constraint Method (Haimes, Hall and Freedman, 1975), and MOLP methods of Steur (1973) and Yu and Zeleny (1975).

The Method of Satisfactory Goals

One of the interactive methods mentioned above, and the one chosen as a basis for solving the WBVRP, is the Method of Satisfactory Goals (Benson, 1975). This method requires the decision maker to identify the 'least satisfactory' achievement level of the K objectives at each iteration. This 'least satisfactory' objective function is then optimized, subject to constraints formed by the original problem constraint set and the acceptable (satisfactory) achievement levels of the remaining objectives. The procedure is rather straightforward, consisting of the following four steps:

- Step 1. Choose feasible satisfactory levels of all objectives.
- Step 2. Choose the least satisfactory achievement level. If none can be identified as least satisfactory, stop. The final solution has been found.
- Step 3. Maintaining other satisfactory achievement levels as constraints, optimize the least satisfactory objective.
- Step 4. If improvement in the least satisfactory achievement level (from Step 3) is not sufficient, revise one or more of the constraining achievement levels and go to Step 3.

Otherwise, the achievement level for the least satisfactory objective may be revised. Go to Step 2.

The starting point in Step 1 is one in which all objective functions have satisfactory achievement levels. This is a solution which is minimally acceptable to the decision maker, but which he/she would

improve upon if possible. In Step 2, the decision maker simply decides which achievement level is farthest from a desired level. If none can be chosen, then all achievement levels are equally satisfactory and the procedure ends with the current solution. Step 3 is the optimization step. The first time this step is performed, the solution is driven toward the nondominated solution set (the beginning solution in Step 1 being feasible but not necessarily nondominated). In subsequent iterations, the solution is moved along the nondominated set. In Step 3, the decision maker must decide whether the least satisfactory achievement level has been sufficiently improved. If not, one or more of the constraining achievement levels must be relaxed in order to allow further improvement to take place. The decision maker is helped in this step by having values of dual variables which indicate, on a local level, the consequences of given levels of constraint relaxation.

The Method of Satisfactory Goals may be applied to linear or non-linear problems, in continuous or integer variables. Figure 3.1 illustrates the method applied to a linear problem in two variables having three objective functions f_1 , f_2 , and f_3 . Problem constraints are C_1 , C_2 , and C_3 , and the nondominated solution set is shown by the heavy lines S_N . In Figure 3.1.a, the decision maker has selected point A as the initial feasible solution. Note that this solution is not a member of S_N . Suppose the decision maker selects f_3 as the objective function with the least satisfactory achievement level. In this case, f_3 is then minimized while the achievement levels of f_1 and f_2 become (inactive) constraints. This results in solution point B, shown in Figure 3.1.b. This solution is a member of S_N . Now, none of the objective functions can be minimized further unless the decision maker is willing to relax at

least one of the other achievement levels. Suppose that the achievement level of objective function f_1 is next chosen as the least satisfactory. Figure 3.1.c shows that the achievement level of objective function f_3 must be relaxed in order for f_1 to be improved. This relaxation results in the feasible subspace defined by the shaded area in Figure 3.1.c. It is obvious that the minimization of f_1 over this feasible subspace will result in solution point C, from which the procedure will continue.

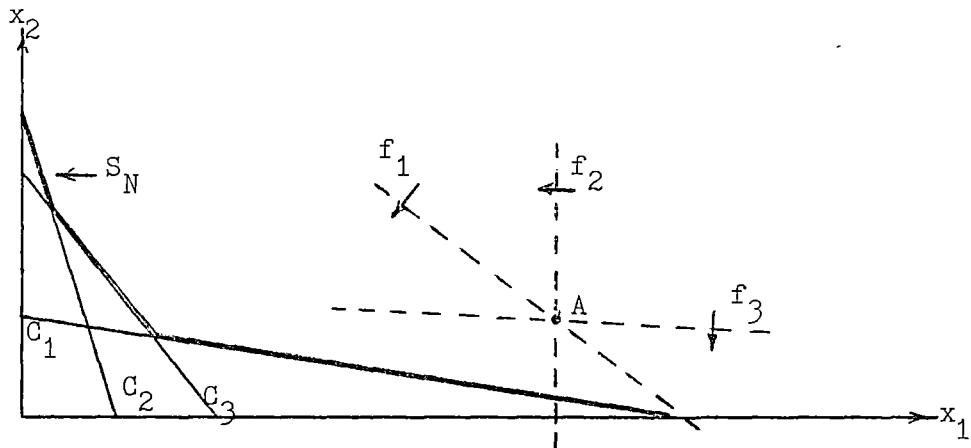
The Method of Satisfactory Goals proceeds in this manner, always optimizing an objective function over a feasible subspace determined by the amount of relaxation applied to the other achievement levels. Because the solution space is a subset of the original feasible space, and because this subspace is bounded in part by specific achievement levels, Benson refers to the solution obtained as 'quasi-efficient'.

The Method of Satisfactory Goals

Applied to the WBVRP

The vector-maximum problem defined by (3.1) :-(3.9) could, in theory, be solved as a multiobjective integer program (MOIP). Some work has been done toward solving multiobjective integer programs; articles by Lee (1977), Zions (1977), Villarreal, Karwan, and Zions (1980), and Klein and Hannan (1982) are representative of research in this area. However, since the underlying VRP is NP-complete, the difficulty of employing such an exact approach to solve problems of reasonable size should be obvious. Recall that the exact approach of Christofides, Mingozzi, and Toth (1981) was used to solve (single objective) VRPs of no more than twenty-five customers.

Faced with the almost impossible task of optimally solving the



a. Initial Satisfactory Feasible Solution

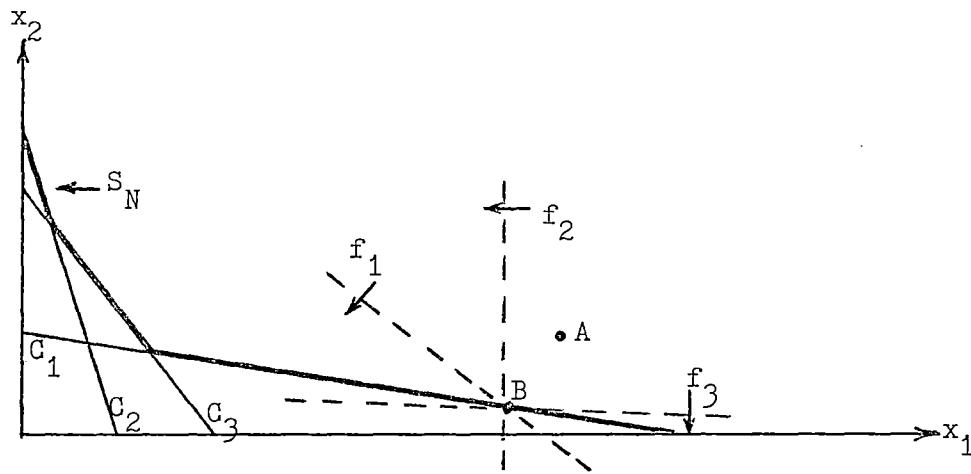
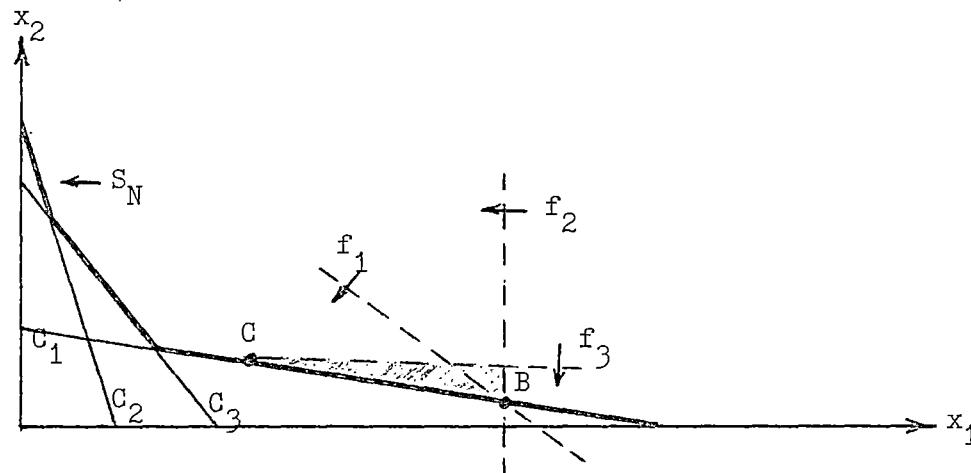
b. Minimization of Objective Function f_3 c. Achievement Level of f_3 Relaxed. f_1 to be Minimized Next.

Figure 3.1. Method of Satisfactory Goals Illustrated

multiobjective VRP, it would seem that a reasonable approach would be to use heuristic methods. Gabbani and Magazine (1985) use an interactive heuristic approach to solving the MOIP. Their approach, based on Steur's method of interval criterion weights (1977), requires that $2K + 1$ single-objective problems be solved and presented to the decision maker at each iteration of the process, where K is the number of objectives in the MOIP. The decision maker is then asked at each iteration to choose his preferred solution from among the $2K + 1$ alternatives. Gabbani and Magazine employ a single objective binary integer program (SOIP) heuristic for each of these $2K + 1$ problems at each iteration. In a series of $(0,1)$ problems solved on an IBM 4341, computation times using this heuristic ranged from 11.72 to 15.78 seconds for an MOIP having three objectives, thirty constraints, and sixty variables. Although extrapolation of these times to a $(0,1)$ problem the size of (3.1) - (3.9) (over 300 variables in more than 3000 constraints for a ten-customer, three-vehicle problem) would no doubt be inaccurate, it should nevertheless be obvious that a general-purpose SOIP heuristic cannot be successfully employed. Use of a special-purpose VRP heuristic (e.g., a 3-opt branch exchange heuristic) to solve each of the seven problems at each iteration would reduce the effort some, but overall computation times would likely still be excessive. Examination of this approach would be a good topic for future research, however.

Even if the solution times of the Gabbani and Magazine approach could be reduced through the use of a different SOIP heuristic, the decision maker still has the task of choosing a preferred solution from among seven different sets of routes at each iteration. This amount of user input, plus the requirement that the decision maker be consistent

with his preferences, seems somewhat excessive. An acceptable alternative, it would seem, would be the use of an interactive satisficing algorithm in which the user expresses his preference for 'satisfactory' levels of objective function achievement. One such approach, the Method of Satisfactory Goals (discussed above), provides this capability. The three objective functions of the WBVRP were given by equations (3.1), (3.2), and (3.3) as before. If f_1 has been designated by the decision maker as the objective function having the least satisfactory achievement level, and if b_2 and b_3 are the previously attained achievement levels of f_2 and f_3 , respectively, then the problem is formulated as

$$\text{Min } f_1 \quad (3.20)$$

$$\text{S. T. } f_2 \leq b_2 \quad (3.21)$$

$$f_3 \leq b_3 \quad (3.22)$$

and (3.4) - (3.9).

Similarly, if f_2 has been designated as the objective function having the least satisfactory achievement level, the problem is defined as

$$\text{Min } f_2 \quad (3.23)$$

$$\text{S. T. } f_1 \leq b_1 \quad (3.24)$$

$$f_3 \leq b_3 \quad (3.25)$$

and (3.4) - (3.9).

Finally if f_3 has been designated as the objective function having the least satisfactory achievement level, the problem is defined as

$$\text{Min } f_3 \quad (3.26)$$

$$\text{S. T. } f_1 \leq b_1 \quad (3.27)$$

$$f_2 \leq b_2 \quad (3.28)$$

and (3.4) - (3.9).

While the general approach of the Method of Satisfactory Goals can be used to solve the WBVRP, the method cannot be used without modification. Due to the fact that the WBVRP cannot be solved optimally, heuristics must be employed to minimize the given objective function at each iteration of the method. This has three implications for the model:

1. The heuristics which are developed for the three objective functions should be efficient (i.e., capable of solving problems within reasonable computing times) and effective (i.e., capable of providing good, albeit non-optimal, solutions).
This issue is covered in Chapter IV.
2. The model should provide the decision maker with guidance concerning the consequences of constraint relaxation at any iteration. The Method of Satisfactory Goals supplies the decision maker with values of dual variables for this purpose.
Heuristic procedures cannot do this, so other means of providing guidance to the decision maker must be investigated. This issue is covered in Chapter V.
3. Heuristics, which cannot guarantee optimality in the single-objective case, cannot guarantee nondominance in the multi-objective case. The decision maker must settle for a 'near efficient' solution (Gabbani and Magazine, 1985). This, too, is discussed in Chapter V.

A flowchart depicting the general WBVRP model structure is shown in Figure 3.2. The procedure begins with the determination of an initial satisfactory route set. This route set is determined using a heuristic distance-minimizing algorithm. Route length deviation and route load deviation are unconstrained. It is assumed that any decision maker is willing to consider a minimum-distance solution 'satisfactory', and that further solutions can proceed from that point. If the decision maker is willing to accept this initial solution without change, the procedure halts. Otherwise, the current value of either route length deviation or route load deviation is chosen as the least satisfactory achievement level, LS. Depending upon the choice of LS, a heuristic length-deviation or load-deviation algorithm is employed to obtain a new solution. This corresponds to the initial execution of Step 3 in the Method of Satisfactory Goals, in which a nondominated solution is sought. From that point on, the model proceeds just as in Steps 2 through 4 on page 62. The procedure ends when no achievement level can be designated as least satisfactory.

When any given solution is displayed, there is a possibility that the decision maker can make a preemptive route adjustment to improve one or more of the routes in the solution. Route adjustments are limited to a single route at a time; i.e., only distance-minimizing adjustments are allowed. Route-length and route-load deviation adjustments are not made preemptively.

Summary

In this chapter, a general model structure to solve the WBVRP has been presented. Assumptions necessary for model development were made,

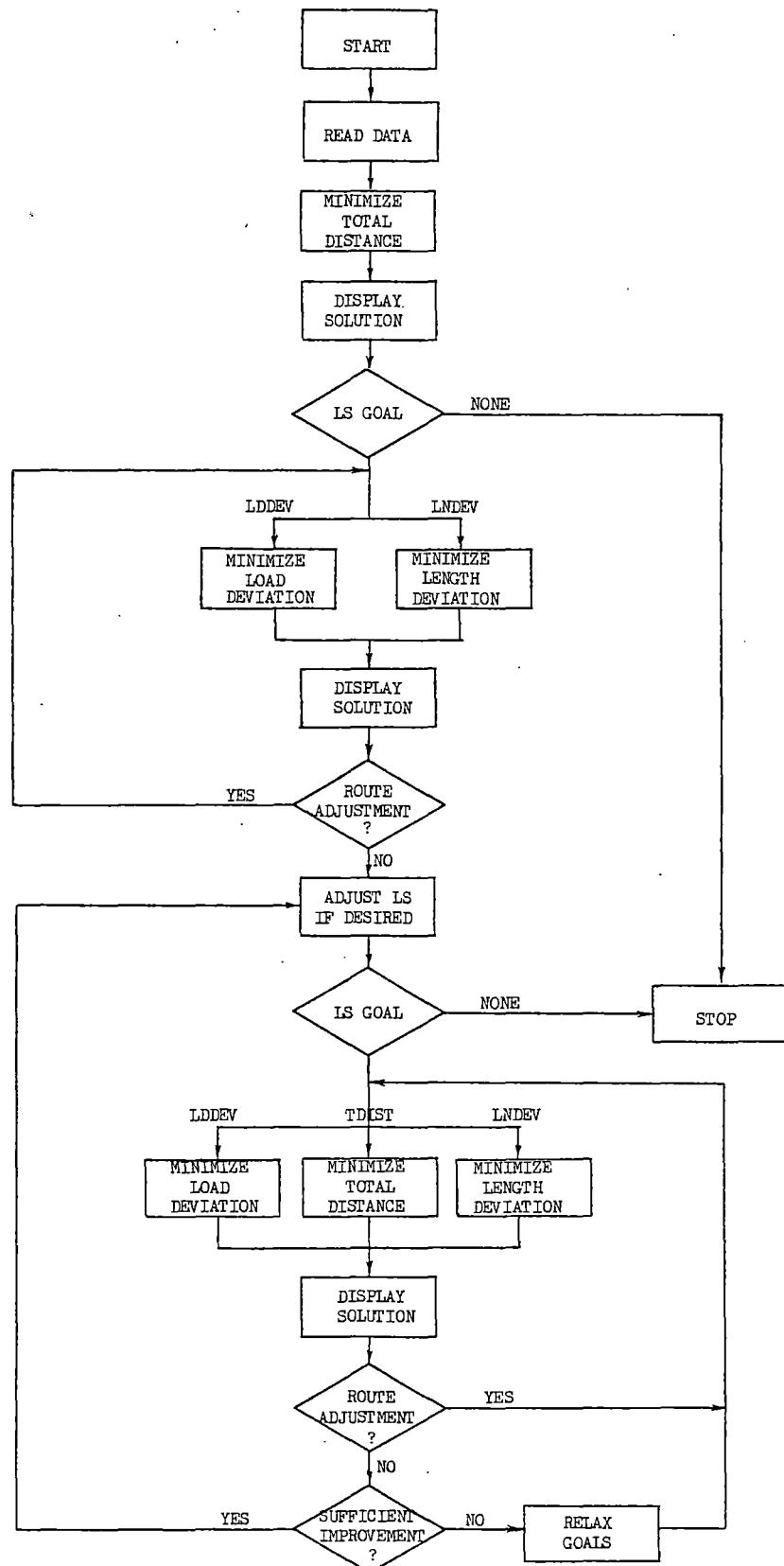


Figure 3.2. General WBVRP Model Structure

and a mathematical formulation of the problem as a vector-maximum (0,1) integer program was given. This was followed by a brief overview of multiple criteria optimization methods. Finally, a heuristic version of one of those methods, the Method of Satisfactory Goals, was chosen as the basis for solving the problem. In this approach, one of three objective functions must be minimized at each iteration, subject to satisfactory achievement levels of the remaining two objective functions. The heuristics employed to minimize these objective functions were not covered here. They are the subject of the next chapter.

CHAPTER IV

SINGLE-OBJECTIVE ALGORITHMS

In the previous chapter, a general model structure for solving the Workload-Balanced Vehicle Routing Problem (WBVRP) was presented. This model involves use of the Method of Satisfactory Goals (Benson, 1975), in which a single objective function is minimized at each iteration of the procedure, subject to satisfactory achievement levels of the other objective functions. The single-objective functions which are to be minimized are:

1. Total Distance,
2. Route Length Deviation, and
3. Route Load Deviation.

Since the WBVRP is NP-complete, the use of heuristics to minimize each of the single-objective functions is necessary. The purpose of this chapter is to develop these heuristic algorithms. For each algorithm, analyses of its effectiveness (ability to produce good solutions) and efficiency (computational speed) are also presented.

Minimization of Total Distance

Minimum-Distance Algorithm

If the total distance is selected by the decision maker as the least satisfactory achievement level, then the total-distance objective function must be minimized, subject to the original problem constraints

and the satisfactory achievement levels of the other two objective functions. The algorithm chosen to minimize total distance is the 3-opt arc-exchange heuristic of Lin (1965) adapted to the VRP by Christofides and Eilon (1969). The algorithm was chosen because of its ability to generate near-optimal solutions. In eight test problems, the 3-opt algorithm produced solutions which were on the average within 1.96 percent of the best solution found in the literature. For details see Cheshire et al. (1982) and Stewart and Golden (1984).

Recall that a solution is said to be k -optimal if it is impossible to improve the solution by removing k arcs from the routes and replacing them with k other arcs. Figure 4.1 shows the different types of arc exchanges which are employed in the 2-opt algorithm and in the 3-opt algorithm. Note that the last three types of 3-opt arc exchanges correspond exactly to the single 2-opt exchange. It is for this reason that any solution which is 3-optimal is also said to be 2-optimal.

The original k -opt algorithm developed for the traveling salesman problem was concerned with only a single route, the TSP tour. The VRP can have up to K routes, where K is the number of vehicles in the problem. In order to use k -opt algorithms for the VRP, it is convenient to restructure the K routes into a single tour. This is done through the use of 'artificial depots'. An artificial depot has the same characteristics as the original depot (i.e., the same coordinates and zero demand), and it is inserted between any two routes in the problem. To form a single tour, $K - 1$ artificial depots are necessary. This is illustrated in Figure 4.2. The dashed lines on either side of the artificial depots in Figure 4.2(B) are used to indicate that the distances are not to scale, the artificial depots having been displaced

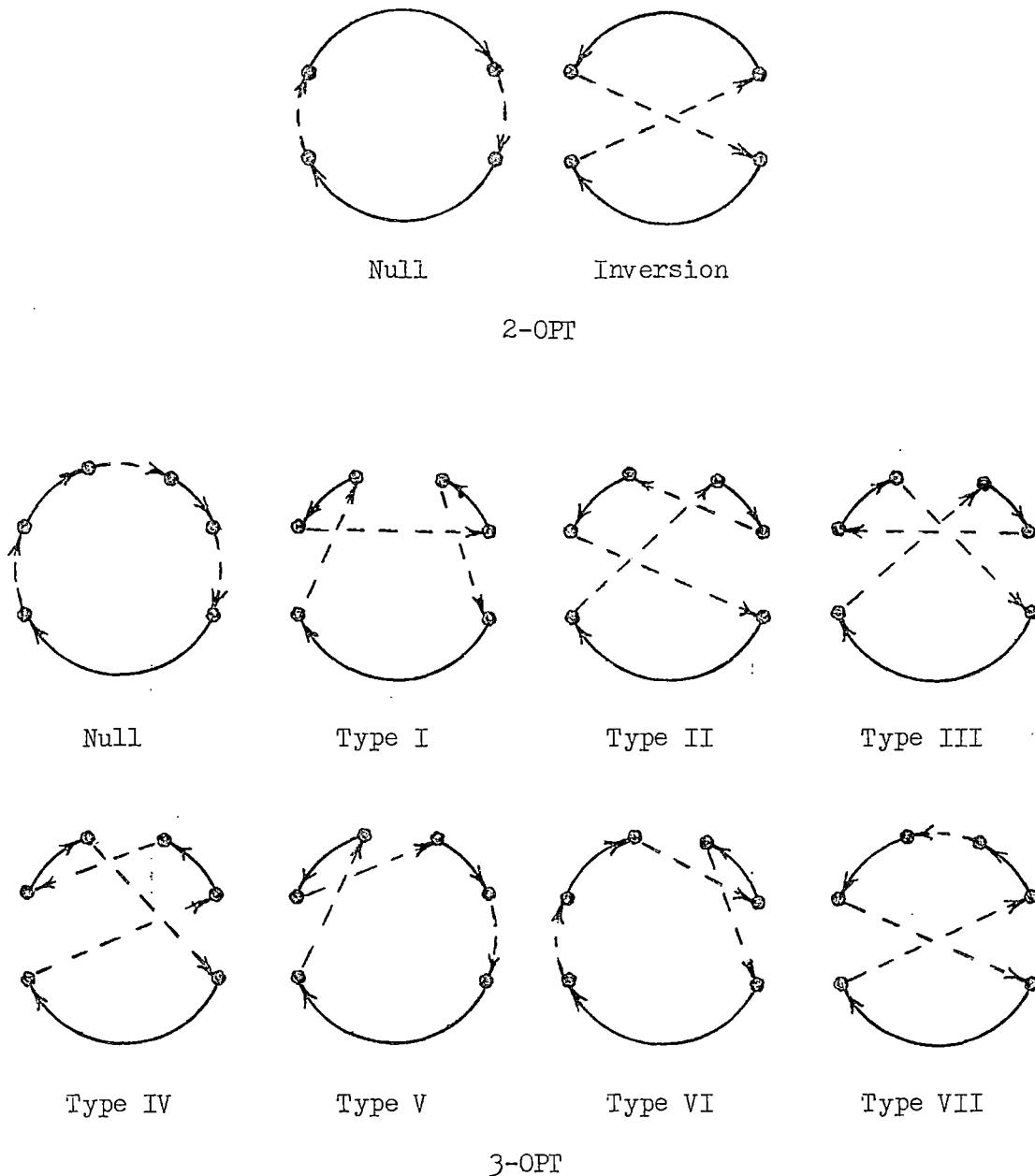
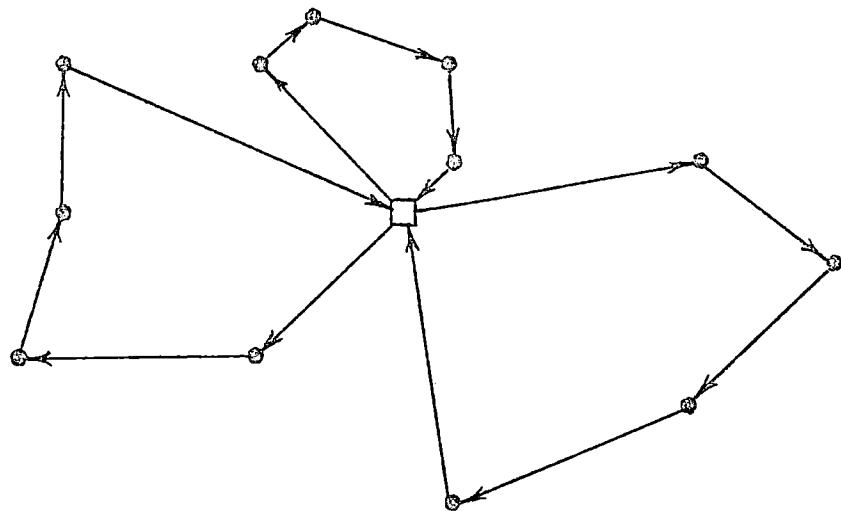
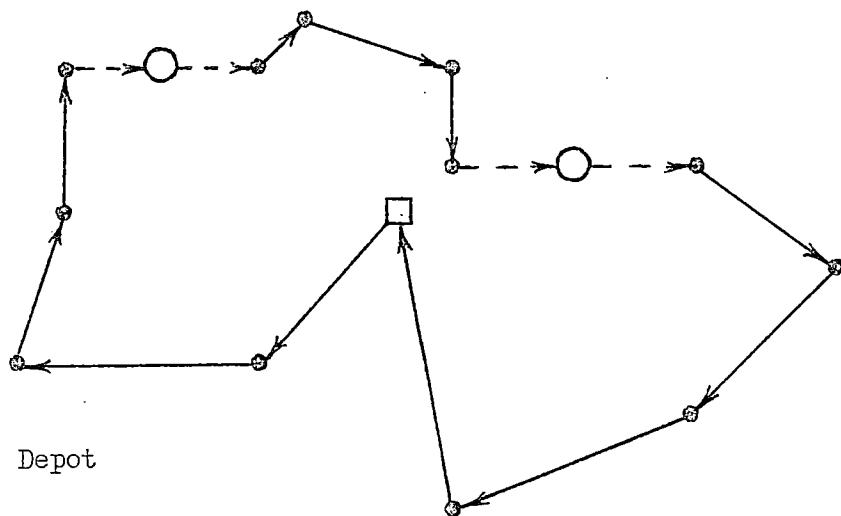


Figure 4.1. 2-opt and 3-opt Arc Exchanges



(A) ORIGINAL ROUTE STRUCTURE



□ Depot

○ Artificial Depot

(B) INSERTION OF ARTIFICIAL DEPOTS

Figure 4.2. Restructuring Routes Using Artificial Depots

from the original depot location in the figure. The depots, original and artificial, are treated just as the other nodes in the k-opt arc exchange process. The only difference is that the resulting routes must be checked for vehicle-capacity and route-length feasibility. In addition, for the WBVRP, the route-length deviation and route-load deviation must be examined to determine that they do not exceed the satisfactory achievement levels of those objectives.

Figure 4.3 is a simplified flow chart of the distance minimizing algorithm. It is valid for either a 2-opt procedure (in which case there is only one type of arc exchange) or a 3-opt procedure (in which case there are seven types of arc exchanges). From a practical standpoint, it is usually more efficient to achieve 2-optimality through the 2-opt procedure before submitting the problem to a 3-opt procedure. This is because the heuristics have a computational difficulty which is a function of N^k , where N is the number of nodes in the network (Golden et al., 1980). The k-opt heuristics have been employed in this manner in the current research.

In Figure 4.3, it can be seen that feasibility checks do not have to be made if all of the selected arcs are in the same route. In this case, the route load, route length, and route-load deviation will remain feasible. The only possible infeasibility could be in route-length deviation, and that caused by an improvement in the (supposed) shortest route. Since route-length deviation is never to be minimized through inefficient routing, the arc exchange must take place even if the route-length deviation constraint is violated.

After an arc exchange is made, the route structure is 'rotated'; i.e., the first arc to be examined in the previous search for a valid

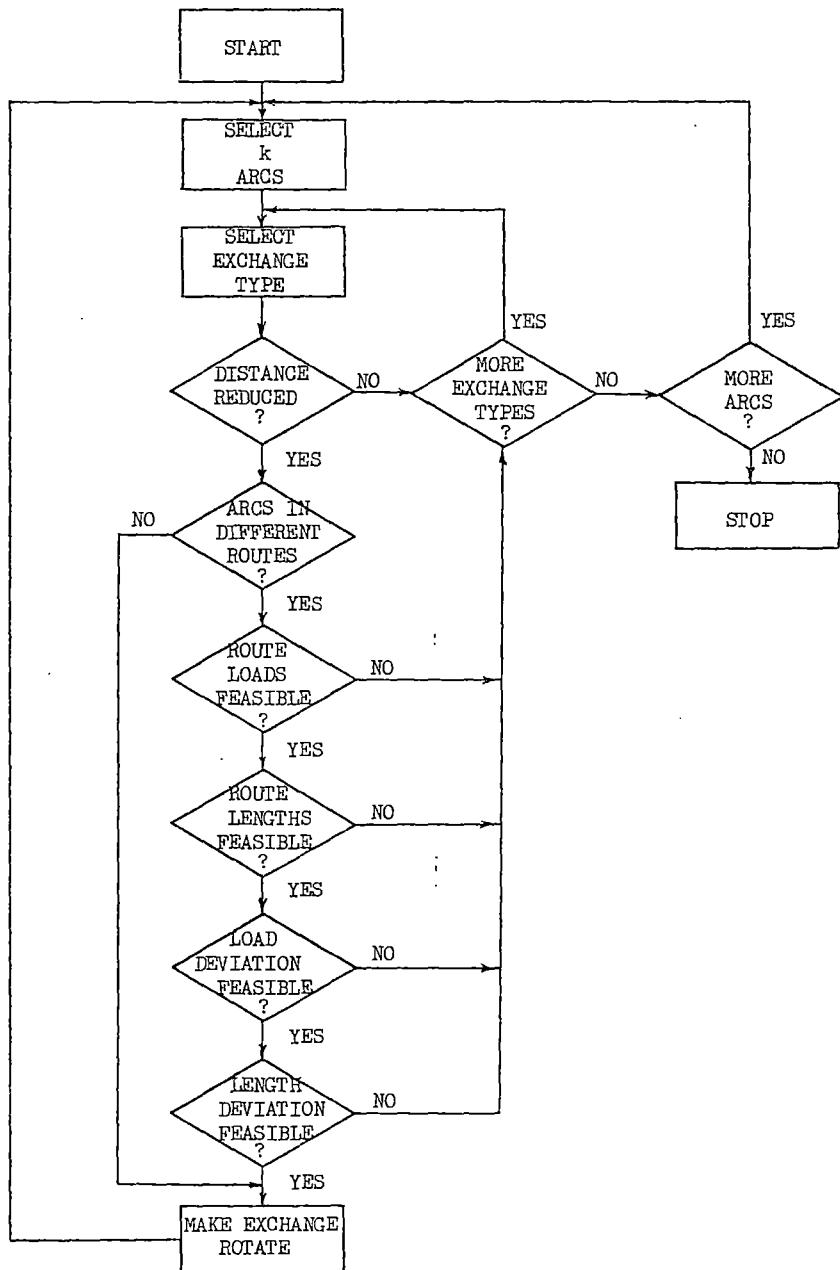


Figure 4.3. Distance Minimizing Algorithm

exchange is replaced in the next search by its predecessor. This is to prevent an exact repetition of the search sequence up to the point of exchange, and increase the likelihood that the next valid exchange will be found quickly. The k-opt procedure ends when all arcs have been examined without finding a valid exchange.

Effectiveness of Algorithm

It is of interest to know how good the answers are which are generated by the distance minimizing algorithm. It was stated above that solutions found by the 3-opt procedure were found to be, on the average, within 1.96 percent of the best solutions published in the literature. However, as is common practice, those 3-opt solutions were found by running the procedure several times (normally ten times) from different starting points and selecting the best one. In an interactive procedure, it is unlikely that this procedure can be followed. It is more likely that restrictions on computing time will limit each distance-minimization subproblem to no more than two or three solutions, from which the best can be selected.

To test the quality of solutions generated by the 3-opt procedure, six problems from the literature and five problems from Chapter VI of this research were selected. The best of one, two, three, and ten runs were selected for each problem and compared with the best known solutions. The best known solutions for the first six problems were taken from the literature, and the best known solutions for the last five problems were found by solving them a minimum of fifteen times each. The solutions and percent errors are shown in Table 4.1. Here, it can be seen that the best of ten runs had an average error of 1.66 percent,

which is close to the 1.96 percent found in the literature. Running the algorithm one, two, and three times and selecting the best solution resulted in an average error of 3.75, 3.17, and 3.14 percent, respectively. These figures will be referred to in Chapter V, in which an interactive computer program employing the distance minimizing algorithm is presented.

Efficiency of Algorithm

The algorithms were all programmed in VS FORTRAN and run on the IBM 3081D at Oklahoma State University. Golden et al. (1980) state that the 3-opt algorithm, when applied to the traveling salesman problem, has a computational difficulty on the order of N^3 , where N is the number of cities visited. It would seem that the computing times for the distance minimization subproblem of the WBVRP would also be a function of N^3 , even though more feasibility checks are necessary in this case (Figure 4.3). To verify this, thirty-six different distance minimization problems were solved using various starting points, and CPU times were determined. The problems selected were Gaskell's 22-city, 29-city, and 32-city problems, and Christofides and Eilon's 50-city, 75-city, and 100-city problems. A model of the form

$$\text{CPU time} = \beta_0 + \beta_1 N^{1/3} \quad (4.1)$$

was chosen, and a regression program was run using the model. Table 4.2 shows the results of the regression. The model obtained from the regression is

$$\text{CPU time (seconds)} = 9.0 \times 10^{-6} + 3.1494 N^{1/3} \quad (4.2)$$

TABLE 4.1
EFFECTIVENESS OF DISTANCE MINIMIZATION ALGORITHM

Problem Number	Source	No. of Cities	Best of n Runs				Best Known Solution	Percent Error			
			n=1	n=2	n=3	n=10		n=1	n=2	n=3	n=10
1	Gaskell (1967)	22	958	958	958	949	949	0.95	0.95	0.95	0.00
2	Gaskell (1967)	29	873	873	873	873	873	0.00	0.00	0.00	0.00
3	Gaskell (1967)	32	809	809	809	809	809	0.00	0.00	0.00	0.00
4	Christofides and Eilon (1969)	50	566	566	566	545	521	8.64	8.64	8.64	4.61
5	Christofides and Eilon (1969)	75	858	851	851	851	845	1.54	0.71	0.71	0.71
6	Christofides and Eilon (1969)	100	880	863	860	860	829	6.15	4.10	3.74	3.74
7	Chapter VI	36	479	473	473	455	426	12.44	11.03	11.03	6.81
8	Chapter VI	36	501	501	501	490	490	2.24	2.24	2.24	0.00
9	Chapter VI	36	546	535	535	535	533	2.44	0.38	0.38	0.38
10	Chapter VI	36	627	627	627	627	627	0.00	0.00	0.00	0.00
11	Chapter VI	36	422	422	422	403	395	6.84	6.84	6.84	2.03
Average error								3.75	3.17	3.14	1.66

The parameters of this model are significant at the 0.01 level, and the model's R^2 is 0.97. Estimated CPU times range from 0.15 seconds for a 22-city problem to 17.91 seconds for a 100-city problem, demonstrating the efficiency of the 3-opt algorithm applied to the WBVRP.

Minimization of Route-Length Deviation

Route-Length Deviation Algorithm

If the decision maker chooses route-length deviation as the least satisfactory achievement level, then the route-length deviation objective function must be minimized, subject to the original problem constraints and the satisfactory achievement levels of the other two objective functions (total distance and route-load deviation). The algorithm developed for this subproblem utilizes arc exchanges similar to the k-opt arc exchanges used to minimize total distance. The heuristic used for route-length deviation is different, however, not only in the objective to be minimized, but also in the set of arcs selected for possible exchanges and in the manner in which the validity of an exchange is determined.

Route balancing methods require a means of clustering customers into potential routes, after which each route length is minimized by solving a TSP. Then the route-length deviation is calculated to determine whether an improvement will result. There are different ways of clustering the customers. Dileepan (1984), for example, uses simple pairwise customer exchanges between two routes at a time. In the current method developed for the WBVRP, 2-opt and 3-opt type exchanges are used for clustering, instead. There are four reasons for using these exchanges for the clustering step:

TABLE 4.2
REGRESSION TABLE FOR 3-OPT SOLUTION TIMES

Source	Degrees of Freedom	Sum of Squares	Mean Square	F Value	P(>F)	R ²	Parameter	Estimate	t Value	P(> t)
Model	1	102.9525	102.9525	1004.41	0.0001	0.97	$\ln(\beta_0)$	-11.6158	-30.51	0.0001
Error	34	3.4855	0.1025				β_1	3.1494	31.69	0.0001
Total (corrected)	35	106.4380								

1. The means of accounting for the different customer exchanges is already provided in the k-opt logic.
2. Many kinds of trades are examined. Arc exchanges can result in one-for-zero, one-for-one, two-for-zero, two-for-one, two-for-two trades, etc., whereas pairwise customer exchanges result only on one-for-one or one-for-zero trades.
3. Exchanges involving up to three routes at a time can be examined.
4. The relative ordering of customers being transferred from one route to another is preserved in the cluster prior to solving the TSP. This is important, since in many cases this same ordering will be optimal in the new route, also.

The TSP for each newly formed cluster is solved heuristically, since the large number of TSPs necessary in the course of solving the route balancing problem precludes the use of exact methods. In the current research, each TSP is solved twice by a 3-opt procedure using two different starting points. If the number of exchanges evaluated for the route balancing problem were the same as the number of exchanges evaluated for the distance minimizing problem, the solving of TSPs by any means would cause the computation times to be prohibitive. Two points are relevant to this concern:

1. The set of arcs which are candidates for an exchange in the route balancing problem is much smaller than the set of arcs in the distance minimization problem.
2. If two routes are involved in a 3-arc exchange, the number of exchange types in the route balancing heuristic is only three, whereas there are seven exchange types in the 3-opt distance minimization heuristic.

The first point is illustrated by considering a typical route balancing problem. In order to improve the measure of imbalance, more than one route must be involved in the arc exchange. All exchanges involving a single route are ignored. In addition, an exchange must include the longest route, the shortest route, or both; otherwise, no improvement in the minimax criterion could be made.

The second point is illustrated by examining Figure 4.4, which shows the clusters which result from a 3-arc exchange involving two routes. Note that arc (2-3), one of the arcs to be eliminated, is in a route to itself. Here it can be seen that the clusters formed by the Type III exchange are the same as those formed by the Type IV exchange. Similarly, the Type I and Type V exchanges result in identical clusters, as do the Type II and Type VII exchanges. Finally, the Type VI exchange is null, since it forms no new clusters. Therefore, only three of the seven exchange types are necessary to provide the route clustering. Exchange types III, V, and VII are selected, since they preserve the relative ordering of customers in the clusters. Similar conclusions would be drawn if arc (5-6) or arc (7-8) had been in a route to itself, although the three exchange types selected would have been different.

Figure 4.5 shows the clusters resulting from a 3-arc exchange involving three different routes. Here, it can be seen that the clusters formed by each type of exchange are different. Therefore, no exchange types can be eliminated in this case.

It should be noted that the 2-arc and 3-arc exchanges do not provide all possible clusterings of customers. All the exchanges except the Type III and Type IV 3-arc exchanges in Figure 4.4 trade contiguous groups of customers beginning with the customer nearest the depot.

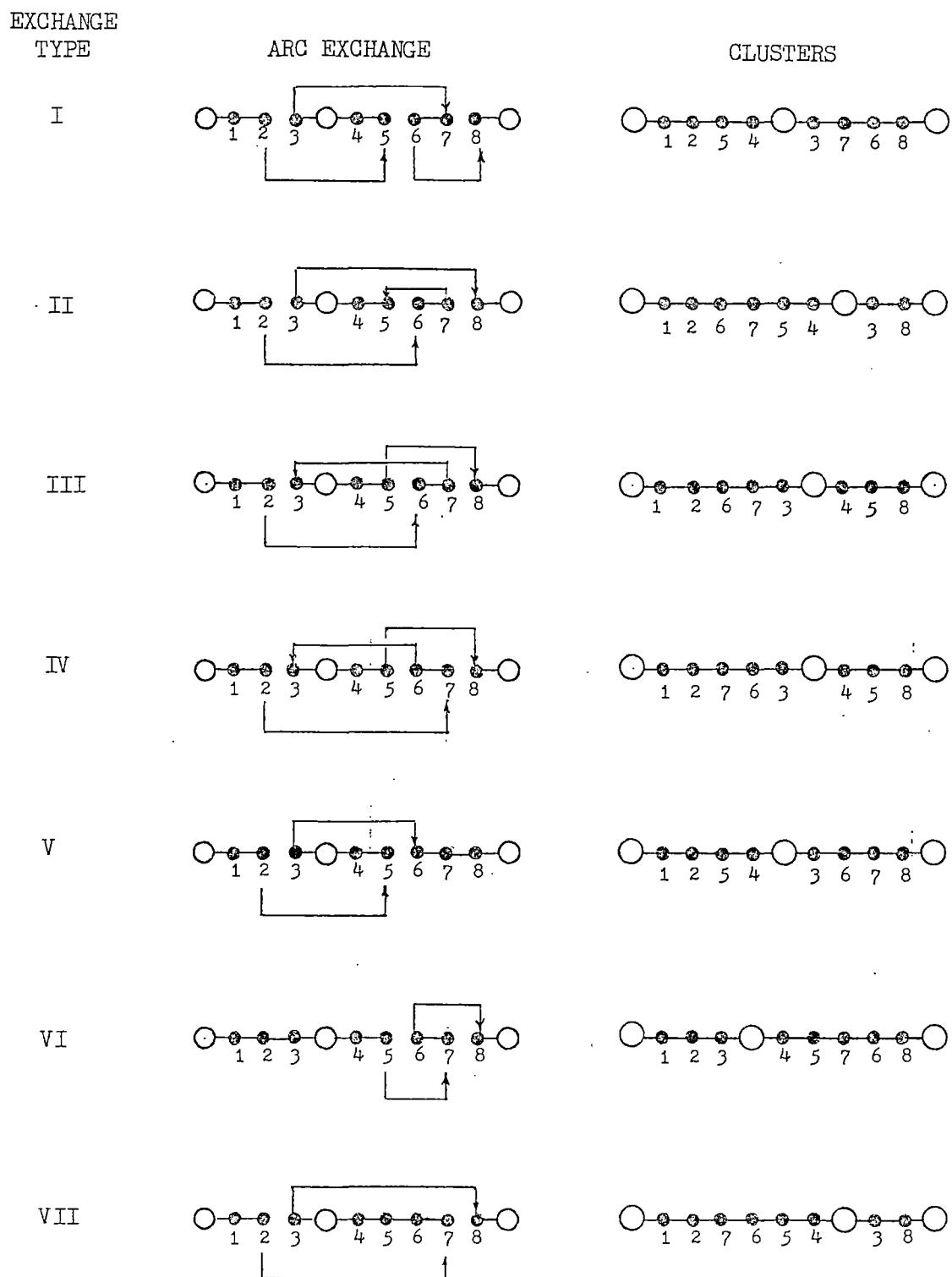


Figure 4.4. Use of Arc Exchanges for Clustering Two Routes

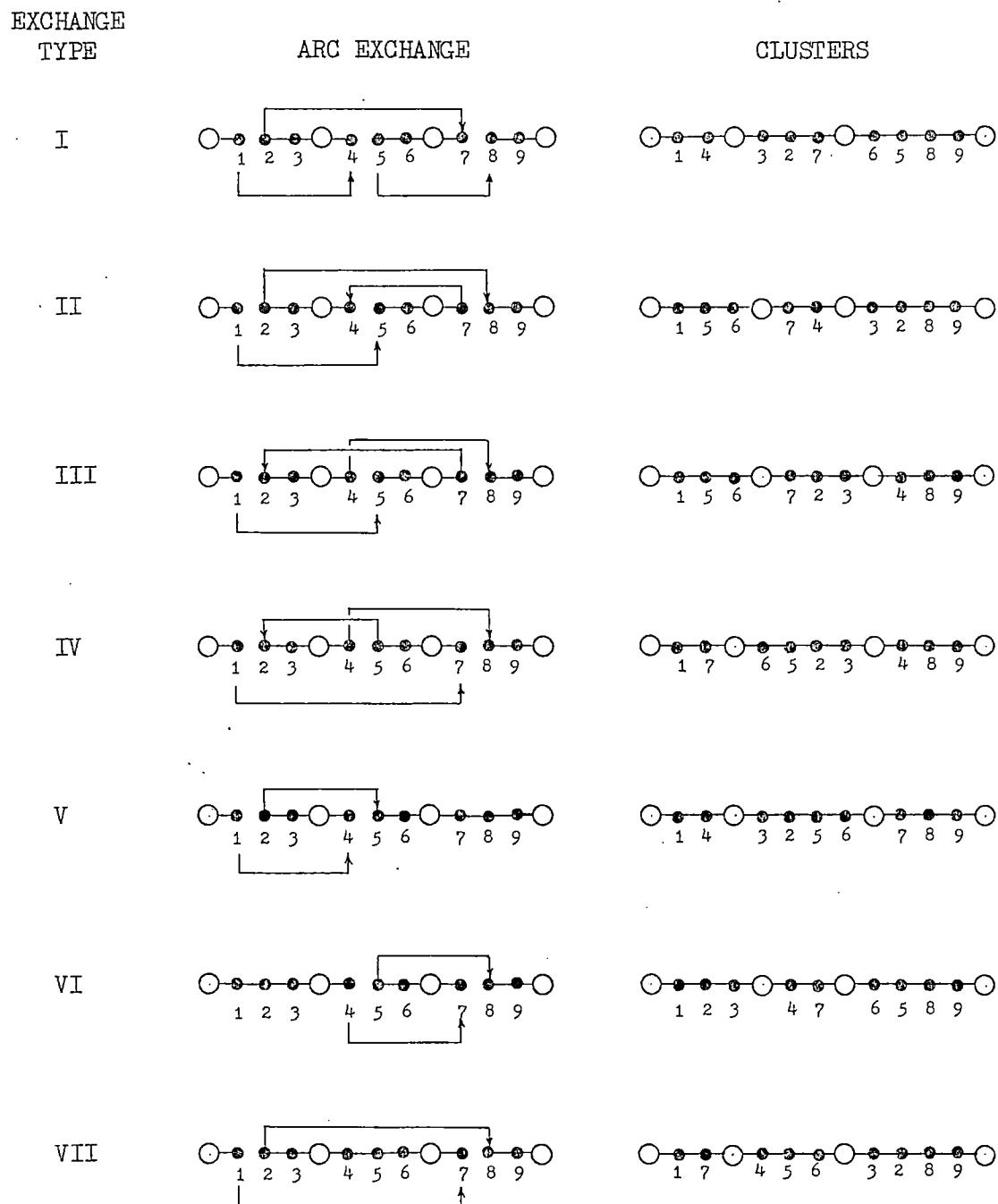


Figure 4.5. Use of Arc Exchanges for Clustering Three Routes

Groups of customers from the middle of each route are not traded in these exchanges. Although this might seem to be a serious limitation on the number of clusters formed, two points should be considered. First, in many cases the most attractive tradeoffs between distance and load involve those customers nearest the depot, since those customers are relatively close to one another. Second, although the immediate result of an exchange does not include trades between the middle of the routes, further exchanges which take place as the algorithm proceeds can produce the same eventual results as if those trades had occurred.

A simplified flow chart for the route-length deviation algorithm is given in Figure 4.6. This flow chart is valid for either 2-arc or 3-arc exchanges. Just as in the total-distance minimization subproblem, the route-length deviation subproblem is solved using 2-arc exchanges prior to being solved using 3-arc exchanges. Whereas the check for improvement in the objective function can be made very early in the total-distance algorithm (see Figure 4.3), this is not possible for the route-length deviation algorithm. TSPs must be solved before an improvement in length deviation can be determined. Because of the computational burden of these TSPs, their solutions are delayed as long as possible in the algorithm. Instead, feasibility checks which can be made without clustering and/or solving TSPs are made in the first part of the algorithm. Any infeasibilities found in these early stages will render the clustering and solving of TSPs unnecessary for the arcs and type of exchange being considered.

Notice that feasibility checks for route loads and route-load deviation can be made without clustering. This is done by keeping track of the cumulative demand, C_i , up to and including customer i , for the

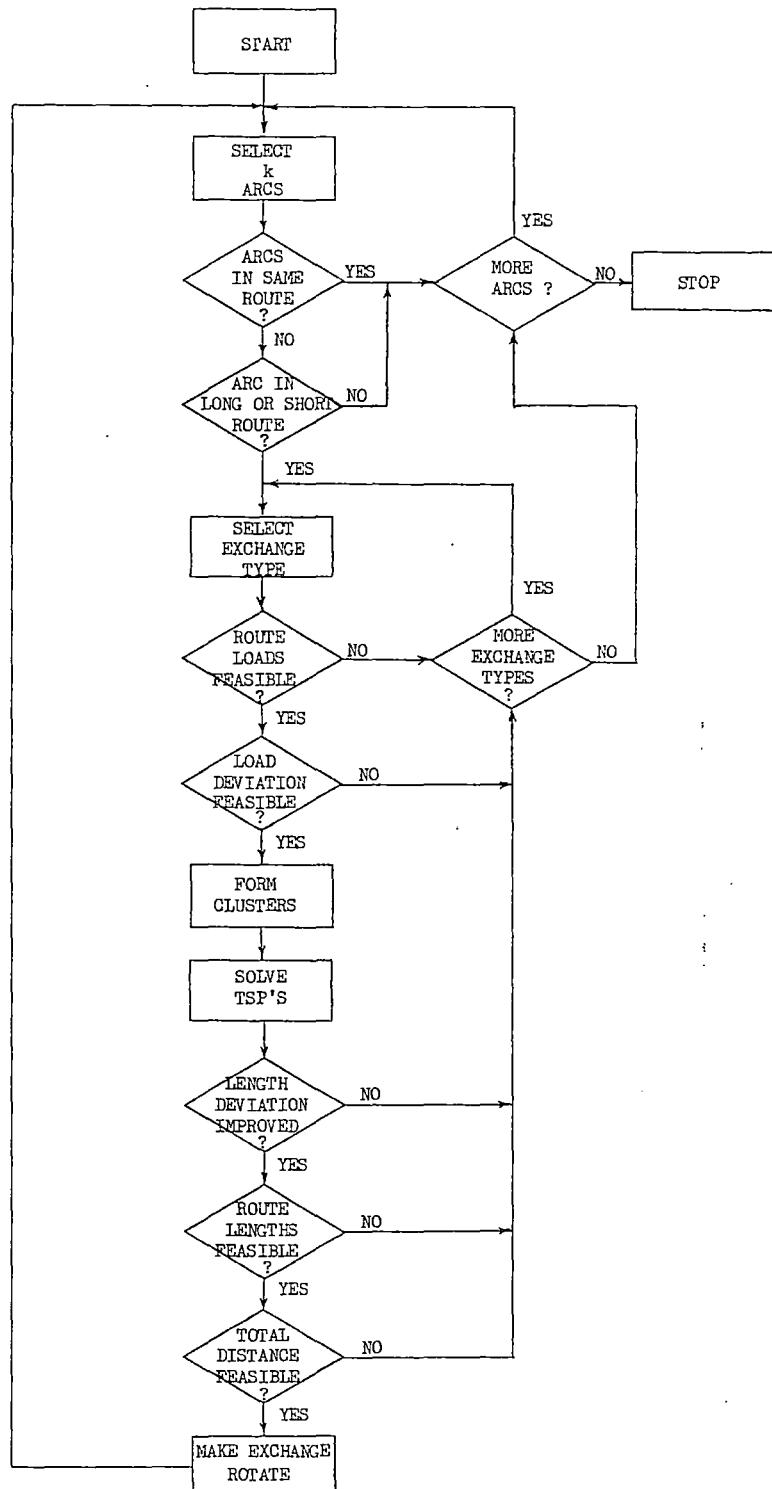


Figure 4.6. Route-Length Deviation Algorithm

route of which i is a member. Consider, for instance, the Type I arc exchange in Figure 4.5. If the current route loads are L_k ($k=1, 2, 3$), then the three new route loads, L'_k , which will result from the Type I exchange involving arcs (1-2), (4-5), and (7-8) are

$$L'_1 = C_1 + C_4 \quad (4.3)$$

$$L'_2 = L_1 - C_1 + C_7 \quad (4.4)$$

$$\text{and } L'_3 = L_2 + L_3 - C_4 - C_7 \quad (4.5)$$

All other route loads in the problem will remain unchanged. Thus, the feasibility checks for route load and route-load deviation can be made without actually making the exchange. Similar rules apply for the other six types of arc exchanges.

Effectiveness of Algorithm

In order to evaluate the quality of solutions obtained from the route-length deviation algorithm, several problems were selected from the literature and solved under different constraining values of total distance, TDIST, and load deviation, LDDEV. The solution chosen as 'best known' in each case was the best of ten runs, obtained by beginning at ten different initial solutions and solving for length deviation, LNDEV. The average percent error measured from this best known solution was 48.83 percent, 28.83 percent, and 18.71 percent for the best of one, two, and three runs, respectively. These percentages seem high in comparison with the errors obtained from the total distance algorithm (see Table 4.1). However, the scale on which the errors are measured has something to do with this. Note that the best known value of LNDEV is less than

fifty distance units in ten of the twelve cases, so a small error in absolute units can result in a relatively high error percentage. Another measure of solution quality is given in the table. This is the percent of possible improvement in the initial value of LNDEV obtained from one, two, and three runs. These are shown to be 91.22 percent, 93.40 percent, and 94.72 percent, respectively. Therefore, the algorithm is seen to be capable of making substantial improvement in route-length deviation beginning with an initial least satisfactory achievement level of this objective.

Efficiency of Algorithm

Because the algorithm requires the solution of many TSPs, and because of the computational burden of solving each such TSP, it would seem that the computer time to solve a route-length deviation problem would be highly correlated with the total number of TSPs solved, as well as their size or complexity. To verify this, the number of TSP solutions required in the course of solving each of forty-one different route-length deviation problems of various size was recorded along with the CPU time required for each problem. A regression analysis was performed on the data using a model of the form

$$\text{CPU} = \beta_0 + \beta_1 \cdot T + \beta_2 \cdot (N/R), \quad (4.6)$$

where CPU = computer time in seconds,

T = Number of TSPs solved in problem,

N = Problem size,

and R = Number of routes in problem.

The quantity (N/R) estimates the average size of TSP solved. The model

TABLE 4.3
EFFECTIVENESS OF ROUTE-LENGTH DEVIATION ALGORITHM

Problem Number	Source	No. of Cities	Constraints		Initial LNDEV	Best of n Runs				Percent Error			Percent Improved		
			TDIST	LDDEV		n=1	n=2	n=3	n=10	n=1	n=2	n=3	n=1	n=2	n=3
1	Gaskell (1967)	22	991	46.47	95	46	46	45	32	43.75	43.75	40.63	77.78	77.78	79.37
2	Gaskell (1967)	22	1001	4351	95	28	28	28	28	0.00	0.00	0.00	100.00	100.00	100.00
3	Gaskell (1967)	22	1004	2945	162	161	161	161	161	0.00	0.00	0.00	100.00	100.00	100.00
4	Gaskell (1967)	32	849	1727	51	21	21	20	12	75.00	75.00	66.67	76.92	76.92	79.49
5	Gaskell (1967)	32	835	1805	51	33	33	33	33	0.00	0.00	0.00	100.00	100.00	100.00
6	Gaskell (1967)	32	951	283	13	13	13	13	13	0.00	0.00	0.00	100.00	100.00	100.00
7	Christofides and Eilon (1969)	50	577	9	10	4	4	4	4	0.00	0.00	0.00	100.00	100.00	100.00
8	Christofides and Eilon (1969)	50	582	11	53	12	10	7	5	140.00	100.00	40.00	85.42	89.58	95.83
9	Christofides and Eilon (1969)	50	725	3	107	17	17	17	11	54.55	54.55	54.55	93.75	93.75	93.75
10	Christofides and Eilon (1969)	75	870	18	97	57	57	57	57	0.00	0.00	0.00	100.00	100.00	100.00
11	Christofides and Eilon (1969)	75	885	18	97	54	54	54	49	10.20	10.20	10.20	89.58	89.58	89.58
12	Christofides and Eilon (1969)	75	1131	5	81	29	13	9	8	262.50	62.50	12.50	71.23	93.15	98.63
Average										48.83	28.83	18.71	91.22	93.40	94.72

was linearized to the form

$$\ln(\text{CPU}) = \ln(\beta_0) + \beta_1 \ln(T) + \beta_2 \ln(N/R). \quad (4.7)$$

The results of the regression are given in Table 4.4. Here a strong relationship between the solution time and number of TSPs is seen, with the parameters all being significant at the 0.01 level, and an R^2 of 0.97 being obtained. The model (4.6) becomes

$$\text{CPU} = 9.2077 \times 10^{-5} \cdot T^{0.9034} \cdot (N/R)^{1.7279}. \quad (4.8)$$

Having shown the relationship between the solution time of a route-length deviation problem and the number of TSPs solved in the problem, it is possible to formulate a model for the solution time of a problem if the number of such TSPs can be estimated. This number is a function of three factors:

1. The number of 'qualified' 3-arc combinations encountered in the course of proving 3-arc optimality; i.e., the number of 3-arc combinations having all arcs not in the same route and having arc(s) in the longest route, the shortest route, or both.
2. The degree to which the workload in the current solution is out of balance.
3. The amount of relaxation in the satisfactory achievement levels of the other two objective functions, total distance and route-load deviation.

The number of 'qualified' 3-arc combinations, Q, can be approximated (see Appendix A) to be

$$Q = \frac{(N+R)^3}{6} - \frac{2(N^3-NR^2)}{6R^3} - \frac{(NR+R^2-2N)^3}{6R^3} \quad (4.9)$$

TABLE 4.4
 REGRESSION TABLE FOR SOLUTION TIME OF ROUTE-
 LENGTH DEVIATION ALGORITHM AS A FUNCTION
 OF NUMBER OF TSP'S SOLVED

Source	Degrees of Freedom	Sum of Squares	Mean Square	F Value	P(>F)	R ²	Parameter	Estimate	t Value	P(> t)
Model	2	17.1912	8.5956	586.49	0.0001	0.97	ln (β_0)	-9.2929	-25.51	0.0001
Error	38	0.5569	0.0147				β_1	0.9034	33.92	0.0001
Total (Corrected)	40	17.7481					β_2	1.7279	18.48	0.0001

The degree to which the current solution is out of balance is given by two quantities:

$$B_{Ld} = \frac{LDDEV}{LD_{Max}} \quad (4.10)$$

and

$$B_{Ln} = \frac{LNDEV}{LN_{Max}} \quad (4.11)$$

where B_{Ld} = Route-load imbalance,

B_{Ln} = Route-length imbalance,

$LDDEV$ = Route-load deviation,

$LNDEV$ = Route-length deviation,

LD_{Max} = Maximum load in route set,

and LN_{Max} = Maximum length in route set.

The degree to which the achievement levels of the other two objective functions are relaxed is given by

$$RLX_{dist} = \frac{DLIMIT - DIST}{DIST} \quad (4.12)$$

$$\text{and } RLX_{Ld} = \frac{LDDVLM - LDDEV}{C - LDDEV} \quad (4.13)$$

where RLX_{dist} = Total-distance relaxation,

RLX_{Ld} = Load-deviation relaxation,

$DLIMIT$ = Limit on total distance,

$DIST$ = Total distance of current solution,

$LDDVLM$ = Limit on route-load deviation,

and C = Vehicle capacity.

A model for estimating CPU time can be formulated as

$$CPU = \beta_0^{\beta_1} \cdot Q^{\beta_2} \cdot B_{Ln}^{\beta_3} \cdot RLX_{dist}^{\beta_4} \cdot RLX_{Ld}^{\beta_5} \cdot (N/R)^{\beta_6} \quad (4.14)$$

This model can be linearized through the use of natural logarithms. A preliminary regression analysis of the linear model showed β_2 to be insignificant. The results of the linear model without β_2 are given in Table 4.5. The parameters are all shown to be significant at the 0.10 level, an R^2 of 0.57 being obtained from the model. The final model becomes

$$\text{CPU} = 2.3485 \times 10^{-4} \cdot Q^{0.9325} \cdot B_{Ld}^{1.3018} \cdot RLX_{dist}^{0.0686} \cdot RLX_{Ld}^{0.0795} \cdot (N/R)^{2.1574} \quad (4.15)$$

CPU times on the IBM 3081D ranged from 0.94 seconds for a problem requiring 315 TSPs to 16.10 seconds for a problem requiring 13,101 TSPs. The average solution time for the set of forty-one problems in the analysis was 5.10 seconds, and the standard deviation was 3.17 seconds.

Minimization of Route-Load Deviation

Route-Load Deviation Algorithm

If the decision maker selects route-load deviation as the least satisfactory achievement level in the WBVRP, then route-load deviation must be minimized, subject to the original problem constraints and satisfactory achievement levels of the other two objective functions. The algorithm to do this is based on arc exchanges, as in the algorithm for route-length deviation. The algorithms differ primarily in the arc set on which exchanges can be made and in the order in which feasibility checks are made. The arc exchanges, which are used to cluster customers into new routes, are the same as shown in Figures 4.4 and 4.5 for the route-length deviation algorithm.

A simplified flow chart of the route-load deviation algorithm is

TABLE 4.5

REGRESSION TABLE FOR SOLUTION TIME OF ROUTE-LENGTH DEVIATION
 ALGORITHM AS A FUNCTION OF PROBLEM SIZE, NUMBER
 OF ROUTES, WORKLOAD IMBALANCE, AND
 ACHIEVEMENT LEVEL RELAXATION

Source	Degrees of Freedom	Sum of Squares	Mean Square	F Value	P(>F)	R ²	Parameter	Estimate	t Value	P(> t)
Model	5	10.1140	2.0228	9.27	0.0001	0.57	$\ln(\beta_0)$	-8.3566	-3.65	0.0009
Error	35	7.6341	0.2181				β_1	0.9235	4.71	0.0001
Total (Corrected)	40	17.7481					β_3	1.3018	4.75	0.0001
							β_4	0.0686	1.94	0.0608
							β_5	0.0795	3.24	0.0027
							β_6	2.1574	3.28	0.0024

shown in Figure 4.7. The similarity with the route-length deviation algorithm can be seen by comparing this flow chart with the one in Figure 4.6. Whereas the clustering step is preceded by a check for load-deviation feasibility in the route-length deviation algorithm, the clustering step is preceded by a check for load-deviation improvement in the route-load deviation algorithm. Under similar conditions, therefore, the route-load deviation algorithm can be expected to require fewer TSP solutions than the route-length deviation algorithm, since an arc exchange is less likely to result in an absolute improvement in a current value of route-load deviation than in a value of route-load deviation which is less than or equal to a relaxed limit on this achievement level. In effect, the feasibility check in Figure 4.7 is a better 'TSP filter' than the improvement check in Figure 4.6.

Effectiveness of Algorithm

To evaluate the quality of solutions produced by the route-load deviation algorithm, several problems were selected from the literature and solved for route-load deviation, LDDEV, under different constraining levels of total distance, TDIST, and route-length deviation, LNDEV. Each problem was solved ten times, and the best of one, two, and three runs was recorded and compared with the best of ten runs (the assumed 'best known' solution). Table 4.6 contains the results of this analysis. The error was found to be 43.07 percent, 37.30 percent, and 24.36 percent for the best of one, two, and three runs, respectively. Just as in the route-length deviation problems shown in Table 4.3, the high error percentages are due in part to the scale involved. For instance, the Christofides and Eilon problems in the table have load-deviations

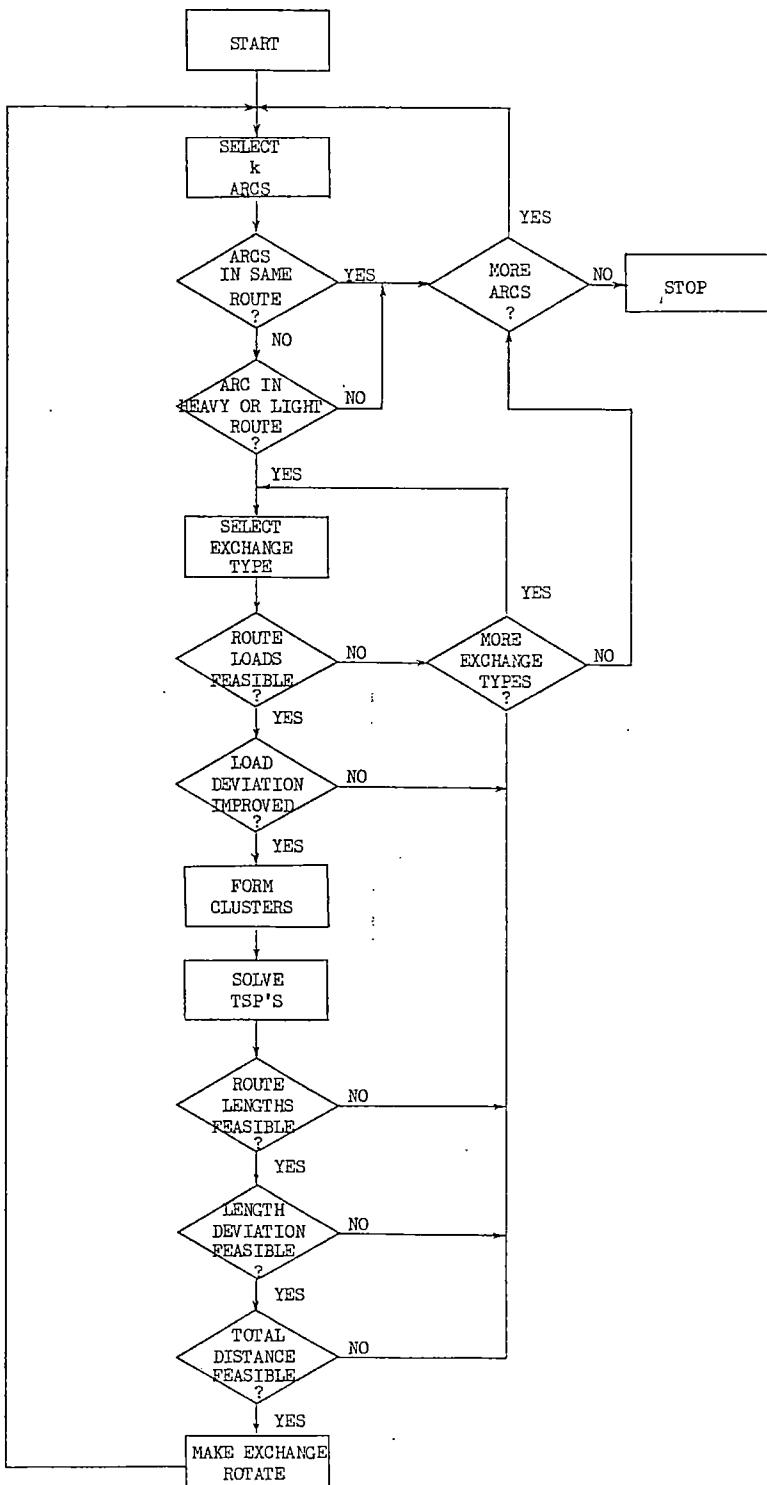


Figure 4.7. Route-Load Deviation Algorithm

TABLE 4.6
EFFECTIVENESS OF ROUTE-LOAD DEVIATION ALGORITHM

Problem Number	Source	No. of Cities	Constraints		Initial LDDEV	Best of n Runs				Percent Error			Percent Improved		
			TDIST	LNDEV		n=1	n=2	n=3	n=10	n=1	n=2	n=3	n=1	n=2	n=3
1	Gaskell (1967)	22	1095	8	3511	3511	3511	3511	3511	0.00	0.00	0.00	100.00	100.00	100.00
2	Gaskell (1967)	22	977	99	4225	3275	3275	3275	3275	0.00	0.00	0.00	100.00	100.00	100.00
3	Gaskell (1967)	22	986	97	4225	3256	3256	3105	3105	4.86	4.86	0.00	86.52	86.52	100.00
4	Gaskell (1967)	32	872	13	1770	750	750	750	750	0.00	0.00	0.00	100.00	100.00	100.00
5	Gaskell (1967)	32	827	53	1570	700	700	700	700	0.00	0.00	0.00	100.00	100.00	100.00
6	Christofides and Eilon (1969)	50	702	3	9	3	3	3	3	0.00	0.00	0.00	100.00	100.00	100.00
7	Christofides and Eilon (1969)	50	586	60	8	1	1	1	1	0.00	0.00	0.00	100.00	100.00	100.00
8	Christofides and Eilon (1969)	50	596	11	7	5	5	5	3	66.67	66.67	66.67	50.00	50.00	50.00
9	Christofides and Eilon (1969)	50	582	6	9	4	4	3	3	33.33	33.33	0.00	83.33	83.33	100.00
10	Christofides and Eilon (1969)	75	919	109	17	7	6	6	4	75.00	50.00	50.00	76.92	84.62	84.62
11	Christofides and Eilon (1969)	75	914	108	17	8	7	6	2	300.00	250.00	200.00	60.00	66.67	73.33
12	Christofides and Eilon (1969)	75	1013	12	10	7	7	5	5	40.00	40.00	0.00	60.00	60.00	100.00
13	Christofides and Eilon (1969)	75	1013	14	10	7	7	5	5	40.00	40.00	0.00	60.00	60.00	100.00
Average										43.07	37.30	24.36	82.83	83.93	92.92

measured in hundredweight units, and the very least error which could occur in these problems is twenty percent, given that there is an error at all. Because of this scaling problem, another measure of solution quality is included in the table. This is the percent improvement made in the initial solution, measured against the possible improvement in going from the initial LDDEV to the best of ten runs. This percent improvement is seen to be 82.83 percent, 83.93 percent, and 92.92 percent for the best of one, two, and three runs, respectively.

Efficiency of Algorithm

As stated above, the check for improvement in route-load deviation in Figure 4.7 provides a better 'TSP filter' than does the check for feasibility of route-load deviation in Figure 4.6. Thus, the solving of fewer TSPs should be required by the route-load deviation algorithm than by the route-length deviation algorithm. This, in turn, should cause the relationship between the total solution time and the number of TSPs solved to be weaker than in the route-length deviation algorithm. To verify this, a regression analysis was performed on data from forty-seven route-load deviation problems using a model of the form

$$\text{CPU} = \beta_0 \cdot T^{\beta_1} \cdot (N/R)^{\beta_2} \quad (4.16)$$

where CPU = computer time in seconds,

T = Number of TSPs solved in problem,

N = Problem size,

and R = Number of routes in problem.

As before, (N/R) is an estimate of the average size of TSP solved. A preliminary analysis on a linear (logarithmic) version of the model showed β_2 to be insignificant. The results of a regression analysis on

a logarithmic model without β_2 is given in Table 4.7. Using these results, equation (4.16) is rewritten as

$$\text{CPU} = 0.3239 T^{0.2028} \quad (4.17)$$

As suspected, the R^2 value of 0.33 is smaller than the R^2 value for the route-length deviation model, which was found to be 0.97 (see Table 4.4). Nonetheless, since the solution time depends to some extent upon the number of TSPs solved, a model similar to equation (4.14) can be written as

$$\text{CPU} = \beta_0 + Q^{\beta_1} \cdot B_{Ld}^{\beta_2} \cdot B_{Ln}^{\beta_3} \cdot RLX_{dist}^{\beta_4} \cdot RLX_{Ln}^{\beta_5} \cdot (N/R)^{\beta_6} \quad (4.18)$$

$$\text{where } RLX_{Ln} = \frac{LNDVLM - LNDEV}{LNDEV} \quad (4.19)$$

and the other terms are the same as defined previously. A regression analysis on a logarithmic version of (4.18) showed only β_0 , β_1 , β_2 , and β_6 to be significant. The results of a final analysis on the logarithmic model are given in Table 4.8. From this table, equation (4.18) can be rewritten as

$$\text{CPU} = 6.8664 \times 10^{-4} + Q^{0.7566} \cdot B_{Ld}^{1.0111} \cdot (N/R)^{1.1964} \quad (4.20)$$

The parameters β_0 , β_1 , β_2 , and β_6 are significant at the 0.05 level, the model obtaining an R^2 of 0.47. Comparing this with the results of the route-length deviation model (Table 4.5), which has an R^2 of 0.57, it is seen that the route-load deviation model is not as capable of explaining the variation in solution times. The CPU times for route-load deviation problems ranged from a low of 0.16 seconds to a high of 5.39 seconds, with an average of 1.30 seconds and a standard deviation of 1.12 seconds.

TABLE 4.7
 REGRESSION TABLE FOR SOLUTION TIME OF ROUTE-LOAD
 DEVIATION ALGORITHM AS A FUNCTION
 OF NUMBER OF TSP'S SOLVED

Source	Degrees of Freedom	Sum of Squares	Mean Square	F Value	P(>F)	R ²	Parameter	Estimate	t Value	P(> t)
Model	1	9.5408	9.5408	22.61	0.0001	0.33	$\ln(\beta_0)$	-1.1273	-4.62	0.0001
Error	45	18.9879	0.4220				β_1	0.2028	4.76	0.0001
Total (corrected)	46									

TABLE 4.8

REGRESSION TABLE FOR SOLUTION TIME OF ROUTE-LOAD DEVIATION
ALGORITHM AS A FUNCTION OF PROBLEM SIZE, NUMBER OF
ROUTES, AND WORKLOAD IMBALANCE

Source	Degrees of Freedom	Sum of Squares	Mean Square	F Value	P(>F)	R ²	Parameter	Estimate	t Value	P(> t)
Model	3	13.3865	4.4622	12.67	0.0001	0.47	$\ln(\beta_0)$	-7.2837	-4.26	0.0001
Error	43	15.1422	0.3521				β_1	0.7566	4.55	0.0001
Total (corrected)	46	28.5287					β_2	1.0111	5.80	0.0001
							β_6	1.1964	2.15	0.0375

Summary

In this chapter three different single-objective algorithms, necessary to the implementation of the Method of Satisfactory Goals in solving the WBVRP, have been presented. These algorithms are used to find minimum values of total distance, route-length deviation, and route-load deviation, each being subject to the original problem constraints and satisfactory achievement levels of the other two objectives. Computational experience was used to provide an evaluation of the effectiveness and efficiency of each algorithm. Tradeoffs between the effectiveness and efficiency of each are necessary in implementing the algorithm in an interactive computer program to solve the WBVRP. This interactive computer program is the subject of the next chapter.

CHAPTER V

INTERACTIVE COMPUTER PROGRAM

Introduction

This chapter contains a description and evaluation of an interactive computer program written to solve the workload-balanced vehicle routing problem (WBVRP). The single-objective algorithms presented in Chapter IV are used in the program to implement a heuristic version of the Method of Satisfactory Goals (Benson, 1975). The program was written in IBM VS FORTRAN and compiled under level two optimization on the IBM 3081D at Oklahoma State University. Graphics display capability is provided by the Tektronix Plot 10 Terminal Control System. All displays illustrated in this chapter are from a Tektronix 4105 graphics display terminal.

Reeves and Franz (1985) list six criteria deserving specific attention in the development of interactive approaches. These are summarized as follows:

1. Minimize required inputs, such as weights or other quantitative assessments, from the decision maker.
2. Simplify the process by, for example, reducing the alternatives presented to the decision maker at each iteration as much as possible.
3. Provide for backtracking, realizing that learning behavior may occur during interaction with the model.
4. Allow the decision maker to reach a satisficing solution in

relatively few steps, realizing that in an interactive process it is not meaningful to expect an exact optimal solution.

5. Structure the choice of alternatives so they are similar at each step, allowing the decision maker to continue using a familiar decision process.
6. Enable the solution of large-scale, real world problems by avoiding methodologies that require the generation of the complete efficient solution set or are otherwise unnecessarily complex computationally.

It is felt that the interactive program described in this chapter meets all of the criteria listed above.

Program Description

Menu

The primary interaction between the decision maker and the program occurs through an on-screen menu which allows the decision maker to specify one of eight different functions. The menu is one of three different displays which appear together on the screen. The other two displays are (1) problem status and (2) tradeoff information, which are illustrated later as a sample problem is solved.

A typical screen display is shown in Figure 5.1. The menu, which appears on the left of the screen, offers eight options. They are:

1. Minimize Total Distance. This option causes a distance minimization problem to be solved, subject to the original problem constraints and satisfactory achievement levels of route-load deviation and route-length deviation (for which the decision maker will subsequently be prompted). Selection of

this option can be the result of a change in the objective function having the least satisfactory achievement level, or can be the result of an insufficient improvement in total distance during the previous iteration.

2. Minimize Load Deviation. A route-load deviation problem is to be solved, subject to the original problem constraints and satisfactory achievement levels of total distance and route-length deviation.
3. Minimize Length Deviation. A route-length deviation problem is to be solved, subject to the original problem constraints and satisfactory achievement levels of total distance and route-load deviation.
4. Manual Route Improvement. The decision maker is to make an adjustment to a route structure by specifying a new ordering of customers in the route. The program evaluates the effects of the change on the route length. If an improvement is made by the adjustment, the changes are implemented; otherwise the manual adjustment is ignored.
5. Display Previous Solution. A prior route structure is displayed graphically, along with the values of each objective function's achievement level. No backtracking occurs if this item is selected.
6. Backtrack to Previous Solution. As in menu item (5), a prior route structure and all of the achievement levels for that solution are displayed. In addition, all of the problem characteristics are reset to the values as contained in that prior solution.

7. Remove Route from Calculations. This option enables one or more routes to be ignored in calculating route-load and route-length deviation. This can be used, for instance, in the case of a very long route serving isolated customers, the inclusion of which would cause unacceptable tradeoffs in attempting to minimize route-length deviation.
8. Exit. The program is terminated, usually after an acceptable solution has been found.

Multiple Runs of Single-Objective Algorithms

Selection of menu items (1), (2), or (3) calls for solution of a single-objective problem. Any arc-exchange algorithm, whether it be a k-opt distance minimization algorithm or a k-arc deviation minimization algorithm, should be run more than once in order to improve the chances that a good solution is reached. To determine the number of runs, a tradeoff between the solution quality and the expense of multiple runs must be reached. The result of the analyses of Chapter IV are used in making this tradeoff. The average CPU time to solve a distance minimization problem (excluding the 100-city problems, for which no solution times were obtained for the other two objectives) was 1.86 seconds, the average time for a route-load deviation problem was 1.30 seconds, and the average time for a route-length deviation problem was 5.10 seconds. The average number of these subproblems required in the solution of a WBVRP was found over a sample of WBVRPs to be almost six. In order to limit the total CPU time to a reasonable amount, the following number of runs for each single-objective algorithm was established for the program:

GASKELL'S 22-CITY PROBLEM SOLUTION NUMBER 3					
MAIN MENU		STATUS	LIMIT	ROUTES:	
				#	LOAD DIST
1.	MINIMIZE TOTAL DISTANCE	--	994	994	1 1144 227
2.	MINIMIZE LOAD DEVIATION	--	3100	****	2 0 0
3.	MINIMIZE LENGTH DEVIATION	--	88	95	3 2400 183
4.	MANUAL ROUTE IMPROVEMENT				4 4225 140
5.	DISPLAY PREVIOUS SOLUTION				5 1295 216
6.	BACKTRACK TO PREV. SOL.				6 1125 228
7.	REMOVE ROUTE FROM CALC.				
8.	EXIT				
ESTIMATED TRADEOFFS:					
		1	2	3	4 5
LOAD DEVIATION	IMPROVEMENT	144	125	41	19 5
TOTAL DISTANCE	RELAXATION	-39	-25	31	21 18
LENGTH DEVIATION	RELAXATION	73	62	0	3 9
ORIGINAL TRADEOFFS	34	REDUCED TRADEOFFS	5		
SELECT FROM MENU					

Figure 5.1. Screen Display Containing Menu, Problem Status, and Tradeoff Information

1. Total Distance: three runs
2. Route-Load Deviation: three runs
3. Route-Length Deviation: two runs

From the results of the single-objective effectiveness analysis (Tables 4.1, 4.3, and 4.6), the following solution quality can be expected from the multiple runs of the algorithms:

1. Total Distance: 3.14% error
2. Route-Load Deviation: 92.92% improvement
3. Route-Length Deviation: 93.40% improvement

Using these criteria, an 'average' six-iteration problem could be expected to require a total of about forty seconds CPU time, not including the time required to reach a beginning solution.

The mechanism for finding alternate starting points for the multiple runs is straightforward. After the first run, the achievement level of the objective function is increased by a factor. The algorithm is then begun, trying to improve this relaxed achievement level. As soon as a feasible arc-exchange is discovered which improves the relaxed achievement level, the route structure is altered and this newly found value of the objective function becomes the value to be improved. In subsequent iterations of the single-objective algorithm, no relaxation of the achievement level is applied. This is seen to be a primal approach, since the problem always remains feasible.

Goal Tradeoffs

In the Method of Satisfactory Goals, the decision maker must select the amount by which one or more goals can be relaxed in order to improve the least satisfactory achievement level (if the last iteration has not

yielded sufficient improvement). Values of dual variables are provided for this purpose. In the heuristic version of the method used to solve the WBVRP, no such values of dual variables can be provided. Some other means of estimating the effects of goal relaxation must be used, instead. The method employed in the interactive computer program involves the calculation of goal tradeoffs during the final 'proving' stage of the single-objective algorithm.

Suppose, in evaluating a particular arc exchange, the exchange is found to be feasible with respect to the original problem constraints but infeasible with respect to one or both of the two constraining achievement levels. The flow charts in Figures 4.3, 4.6, and 4.7 indicate that the evaluation of that particular arc exchange would be abandoned and another set of arcs selected for evaluation. To provide tradeoff information, however, the evaluation is not abandoned, but continues as if the arc exchange were feasible. As a final step, the program calculates the amount of goal relaxation required to make the exchange feasible. The amount of improvement in the objective function and the amount of relaxation in the constraining achievement levels are stored as a single tradeoff, along with the arcs involved in the exchange. All such tradeoffs are kept in an array for display to the decision maker. The array is reduced by eliminating any dominated exchanges (those exchanges providing lesser improvements in the objective function for greater relaxation in the constraints). The reduced (nondominated) tradeoffs are sorted in order of decending amount of objective function improvement and displayed to the decision maker, up to a maximum of six such tradeoffs. A typical set of such tradeoffs is shown in the lower portion of Figure 5.1.

A tradeoff will require the relaxation of one or both of the two constraining achievement levels. If both relaxations indicated in a given tradeoff have positive values, then both constraints will be increased. If one of the relaxations is negative, the immediate result of the arc exchange is an improvement in that particular achievement level. However, that constraint is not tightened by the program, since the Method of Satisfactory Goals does not operate by tightening constraints. The constraint on the achievement level showing a negative relaxation actually remains unchanged during the solution of the subsequent single-objective problem. The decision maker must realize this when choosing a tradeoff. Only positive relaxations should be used in making this choice; negative relaxations should be considered only when trying to decide between two or more tradeoffs which are otherwise equally attractive.

The tradeoffs provided by the program are only local estimates of the effects of constraint relaxation. After an arc exchange corresponding to the tradeoff is made, the algorithm attempts to make further improvements in the objective function using the new achievement level constraints. If further improvements can be made, then the effect of the tradeoff has been underestimated by the tradeoff. Table 5.1 contains the results of various tradeoffs made in the course of solving 28 different problems. The table shows the percent improvement 'promised' by the tradeoff, the percent improvement actually obtained, and the ratio of the two, expressed as an 'improvement ratio'. Of the 28 tradeoffs in the table, 13 resulted in actual improvements which were the same as promised by the tradeoffs. The remainder resulted in improvements better than indicated by the tradeoffs. The average improvement ratio from the problems in Table 5.1 is 1.35, and the standard deviation is 0.67. Now, it

TABLE 5.1

ACTUAL VERSUS PROMISED IMPROVEMENT IN OBJECTIVE
FUNCTION PROVIDED BY TRADEOFFS

Problem Number	No. of Cities	Objective Function	Percent TDIST	Constraint Relaxation	Percent Promised Improvement	Percent Actual Improvement	Improvement Ratio
		TDIST LNDEV LDDEV	TDIST	LNDEV	LDDEV		
1	22	X		62.00	2.74	2.74	1.00
2	22	X		5.38	0.21	0.31	1.48
3	22	X	11.28		60.00	90.53	1.51
4	22	X		350.00	3.28	5.21	1.59
5	22	X		55.56	2.50	1.02	1.00
6	22	X	7.92		10.87	13.62	1.25
7	22	X		245.00	5.87	5.87	1.00
8	32	X	3.33	11.76	50.32	50.32	1.00
9	32	X	1.44		65.38	65.38	1.00
10	32	X	0.24	15.91	14.81	25.93	1.75
11	32	X		650.00	3.65	4.43	1.21
12	32	X	0.55	80.00	51.18	51.18	1.00
13	32	X		175.00	34.00	34.00	1.00
14	32	X		9.62	70.00	0.23	1.00
15	50	X	3.28		37.50	18.52	1.50
16	50	X	12.54		54.55	86.36	1.58
17	50	X	0.34		140.00	35.29	1.00
18	50	X	2.90			45.45	1.00
19	50	X	1.00		25.00	16.67	2.00
20	50	X		75.00		42.86	1.33
21	50	X		171.43		33.33	1.00
22	75	X	3.53	4.35		37.50	1.50
23	75	X	3.68			20.00	1.00
24	75	X			125.00	3.15	1.18
25	75	X		3.19	11.11	0.12	1.00
26	75	X	6.79			43.62	54.26
27	75	X	3.18		55.56	46.51	1.24
28	75	X	1.38			10.00	53.49
							1.15
							4.50

would be desirable to have a small standard deviation in the improvement ratio. Then the decision maker could be confident that the problem is progressing in the right direction. A large standard deviation increases the chances that the decision maker will choose a tradeoff which results in a solution less than otherwise desirable; that is, a tradeoff not selected could have resulted in greater improvement.

Flow Chart

Figure 5.2 shows a simplified flow chart of the interactive program's main routine. The following variables are used in this flow chart:

DLIMIT = Limit (constraint) on total distance.
LDDEV = Route-load deviation.
LDDVLM = Limit on route-load deviation.
LNDEV = Route-length deviation.
LNDVLM = Limit on route-length deviation.
OBJ = Objective function solved in the previous iteration:
 (1). Total distance
 (2). Route-load deviation
 (3). Route-length deviation
TDIST = Total distance (sum of all route lengths).

Only those subroutines called directly by the main program are shown in Figure 5.2. These subroutines (excluding utility functions and Plot 10 graphics routines) perform the following functions:

ADJUST - Accepts decision maker's manual route adjustments to the route structure, and evaluates the effects of those adjustments.

BKTRAK - Backtracks to a specified prior solution.

DISPLA - Displays (only) a specified prior solution.

LDDV2 - Minimizes route-load deviation using 2-arc exchanges.

LDDV3 - Minimizes route-load deviation using 3-arc exchanges.

LNDV2 - Minimizes route-length deviation using 2-arc exchanges.

LNDV3 - Minimizes route-length deviation using 3-arc exchanges.

LOCK - Excludes (locks out) one or more routes in calculation of route-load deviation and route-length deviation.

NONDOM - Reduces set of tradeoffs by eliminating dominated tradeoffs.

SAVNGS - Minimizes total distance using Clarke and Wright's savings algorithm.

TWOOPT - Minimizes total distance using a 2-opt arc exchange algorithm.

THROPT - Minimizes total distance using a 3-opt arc exchange algorithm.

The main program depicted in Figure 5.2 represents a computer implementation of the general WBVRP model structure of Figure 3.2. Note that the first solution presented to the decision maker (first page of Figure 5.2) is the best of eight total distance minimization solutions, those solutions being obtained by successive implementations of the Clarke and Wright savings algorithm, the 2-opt algorithm, and the 3-opt algorithm. Alternate starting solutions are obtained by randomly mixing up the order of the first few elements of the savings file (those elements responsible for initial route formation). This initial solution is the only one which is chosen from so many iterations of a single-objective algorithm. The reason for doing so is to try to reach a

satisfactory solution to begin with, an assumption being that the decision maker will consider a minimum-distance solution a satisfactory starting point. After this initial solution, the decision maker is given only three choices: (1) accept the solution as a final one, (2) minimize route-load deviation, or (3) minimize route-length deviation. The minimization of total distance is not a choice here, since it is assumed that a minimum distance solution has been obtained by the eight iterations of the distance minimization algorithms. After this, the program will always return to the menu display on the second page of the flow chart, ultimately terminating when the decision maker selects the last menu option.

Evaluation of Interactive Computer Program

To evaluate the performance of the interactive program, two areas are considered. First is the effectiveness of the procedure, or its ability to generate good solutions. Second it is efficiency, or time required to reach a final solution.

Effectiveness of Program

Convergence Analysis. One of the ways of evaluating the effectiveness of a procedure is to determine whether the procedure will converge to the same final solution from different starting points. In order to demonstrate this, some means of insuring consistency on the part of the decision maker must be provided. To do this, a linear additive utility function is assumed. Recall that the Method of Satisfactory Goals does not assume any type of utility function; the only reason for using one here is to provide an objective means of evaluating the

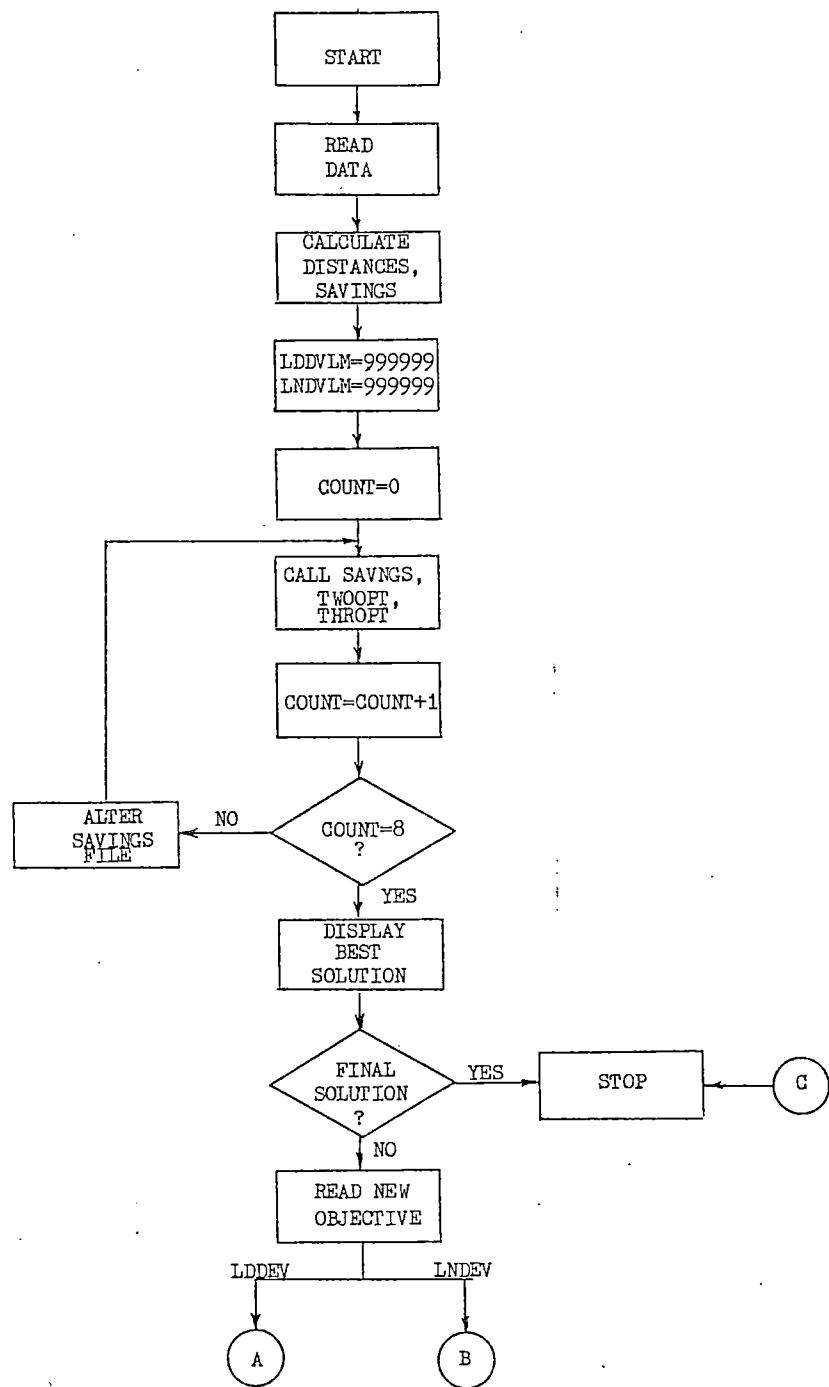


Figure 5.2. Flow Chart for Main Program

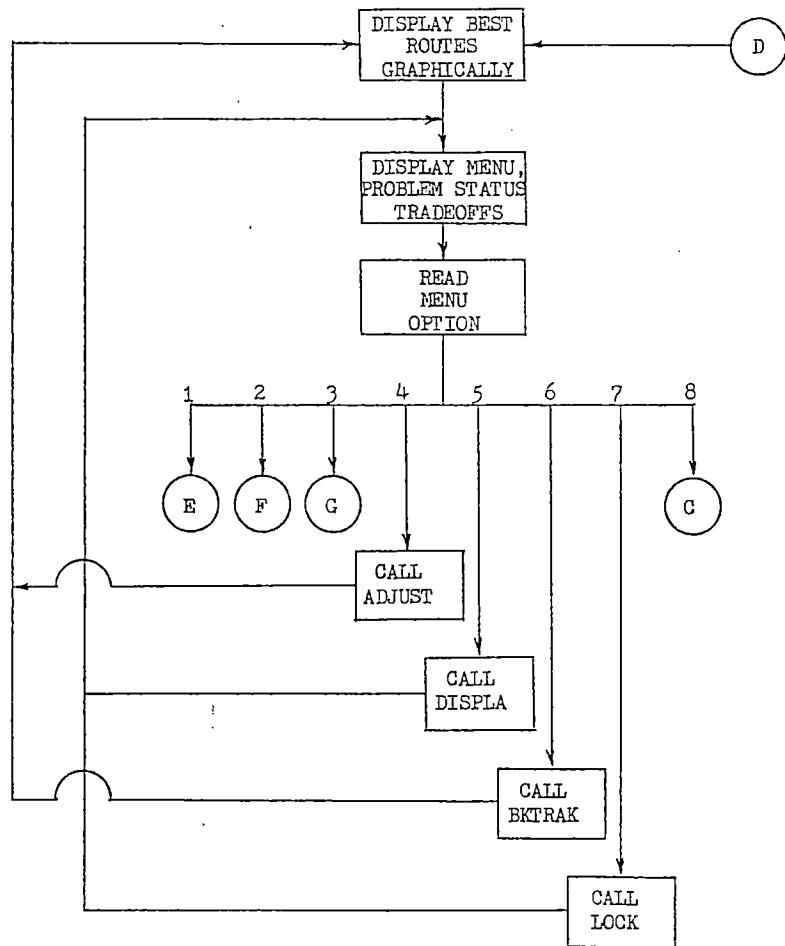


Figure 5.2. (Continued)

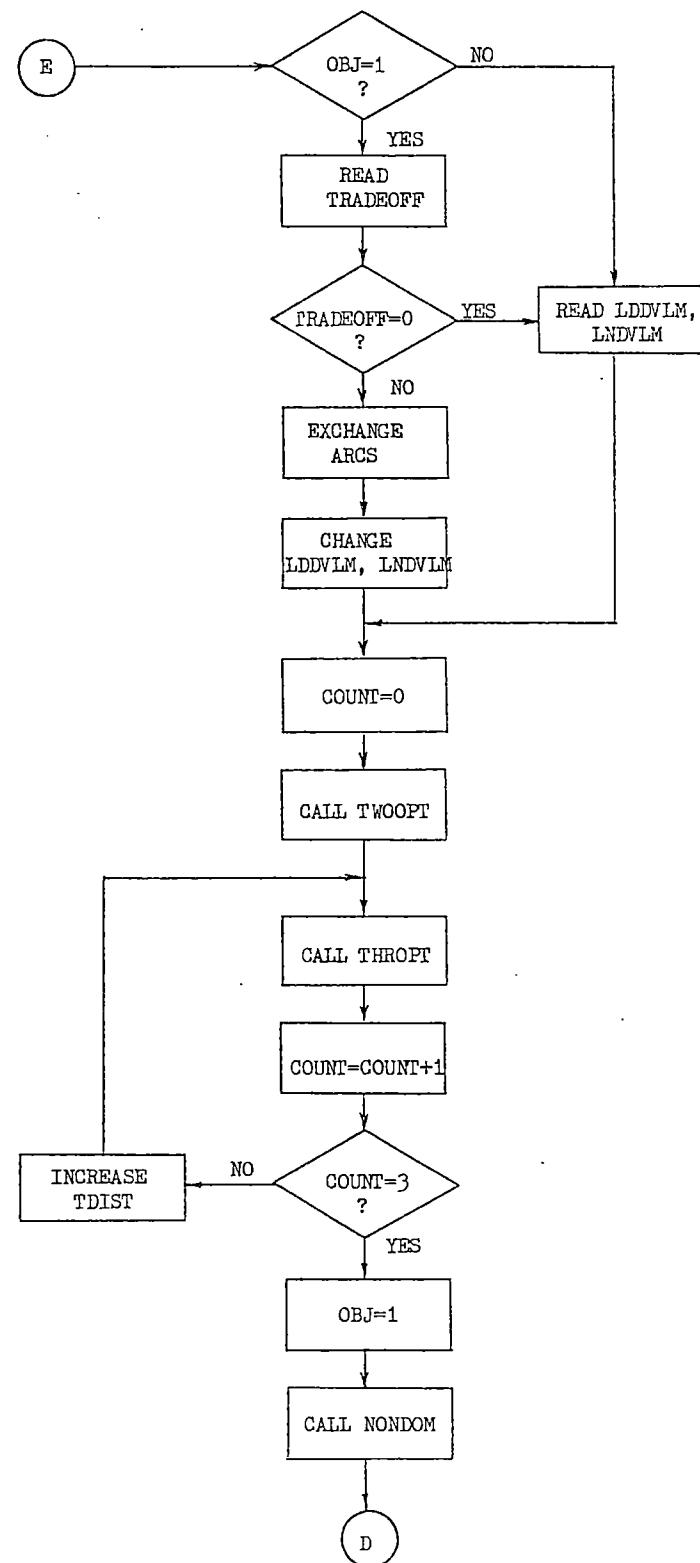


Figure 5.2. (Continued)

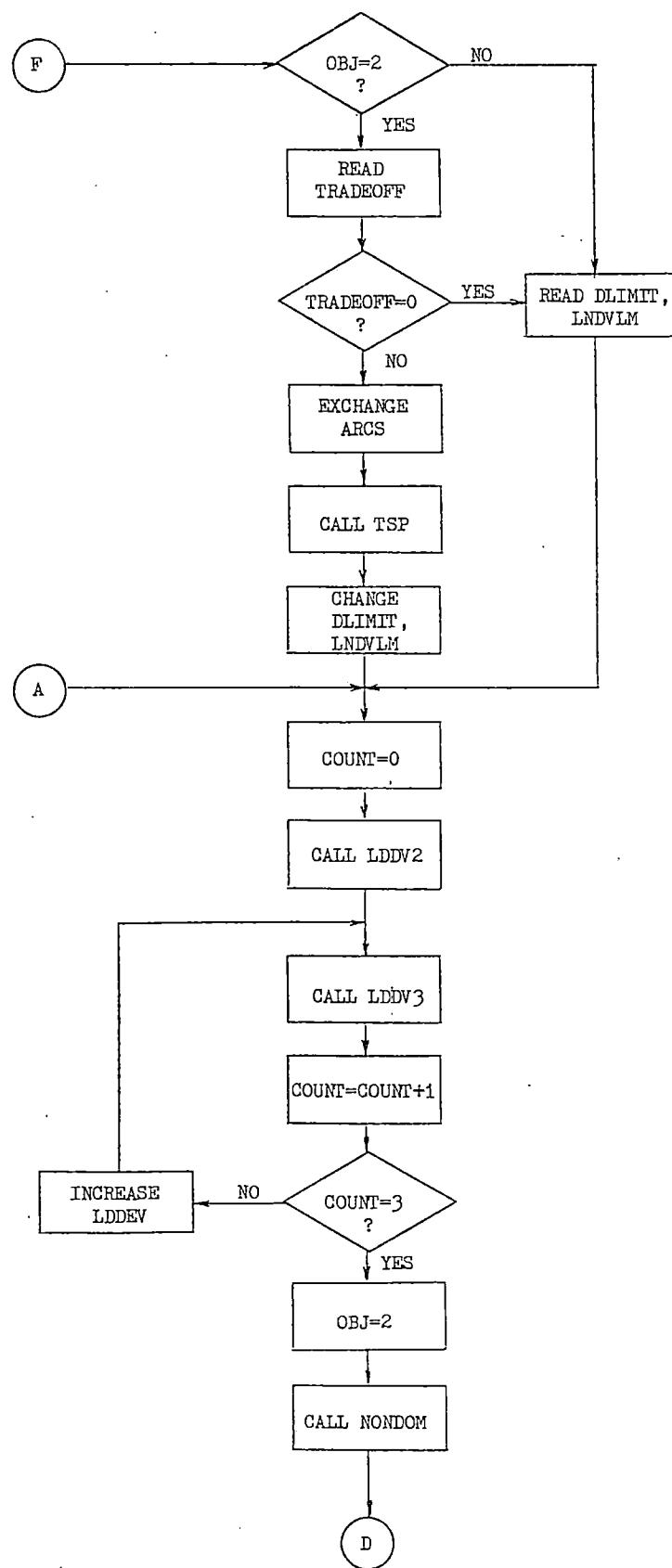


Figure 5.2. (Continued)

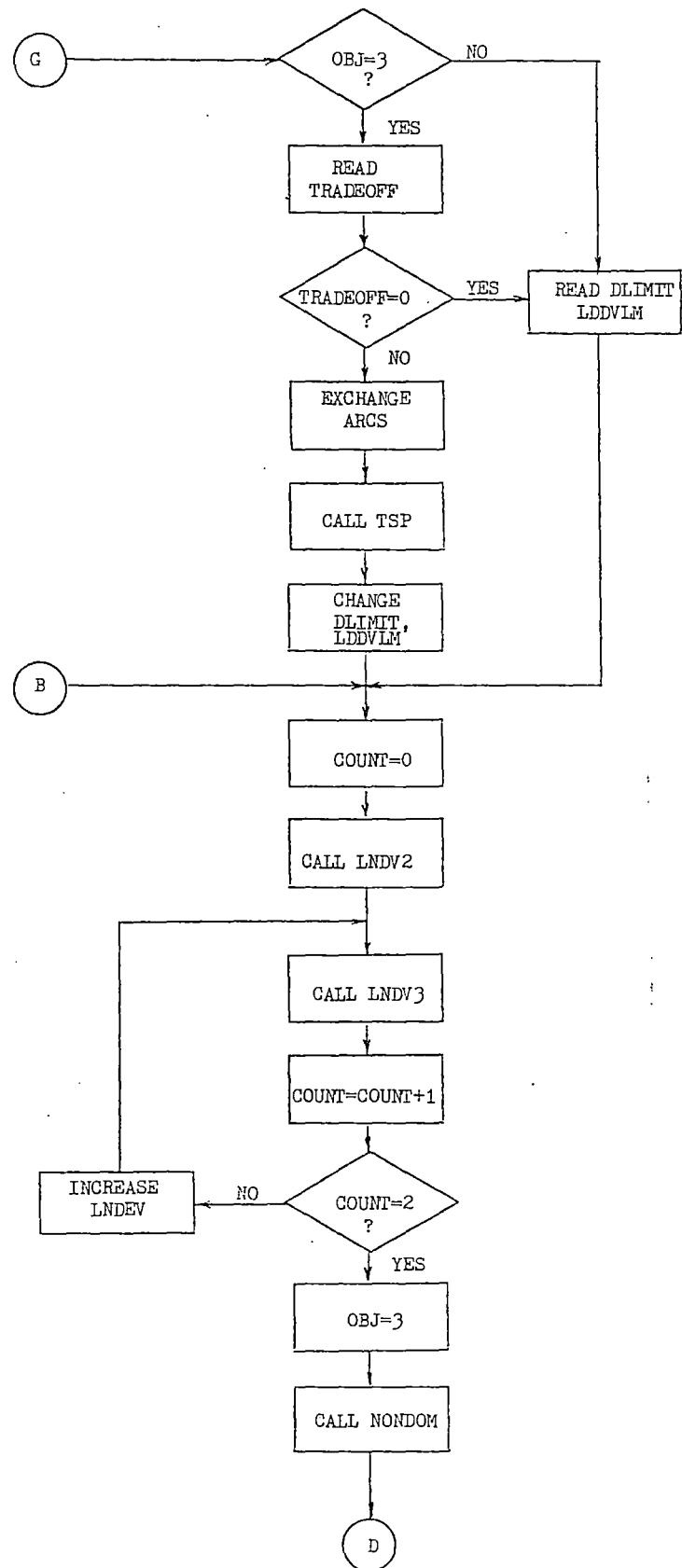


Figure 5.2. (Continued)

interactive procedure, free from inconsistencies introduced by the decision maker's choices.

To formulate the utility function, the range for each objective function is first scaled. For the i th objective, call the best (Ideal) solution value obtained for that objective I_i , and call its worst (Anti-Ideal) solution value A_i . The range of values in going from that objective's worst solution value to its best solution value is

$$R_i = A_i - I_i \quad (5.1)$$

A scaling factor, α_i , is applied to each objective function. Let α_1 be applied to total distance, α_2 to route-load deviation, and α_3 to route-length deviation. The scaling factors are found by simultaneously solving the following three equations:

$$\alpha_1 R_1 - \alpha_2 R_2 = 0, \quad (5.2)$$

$$\alpha_2 R_2 - \alpha_3 R_3 = 0, \quad (5.3)$$

$$\text{and } \alpha_1 + \alpha_2 + \alpha_3 = 1. \quad (5.4)$$

For an unequally weighted utility function (e.g., a 3-2-1 weighting on the three objectives), α_1 is replaced by $3 \times \alpha_1$, α_2 is replaced by $2 \times \alpha_2$, and the new weights are again normalized to sum to unity. The utility function is written

$$U = \alpha_1 \cdot TDIST + \alpha_2 \cdot LDDEV + \alpha_3 \cdot LNDEV. \quad (5.5)$$

To determine the least satisfactory achievement level for purposes of the analysis, the current achievement level is measured against the Ideal solution for each objective function. Call this the measure of satisfaction, S_i , of the i th objective:

$$S_1 = \alpha_1 (TDIST - I_1), \quad (5.6)$$

$$S_2 = \alpha_2 (LDDEV - I_2), \quad (5.7)$$

$$\text{and } S_3 = \alpha_3 (LNDEV - I_3). \quad (5.8)$$

The objective having the greatest value of S_i is selected as the least satisfactory achievement level.

To determine which tradeoff to accept, given that the least satisfactory achievement level has not been sufficiently improved in the last iteration, the potential improvement in the utility function, ΔU , is evaluated for each tradeoff presented to the decision maker. Thus, for the total distance objective,

$$\Delta U = \alpha_1 T_1 - \alpha_2 T_2 - \alpha_3 T_3, \quad (5.9)$$

where T_i is the tradeoff quantity displayed for the i th objective function. Any negative tradeoff quantity is not used in the calculation, since only positive achievement level relaxations are used in the subsequent single-objective problem, as explained in 'Goal Tradeoffs'.

Table 5.2 contains the results of fifteen problems using five different utility functions. For each utility function, three different starting points are used to obtain a final solution. The last column shows the percent error, measured against the minimum value obtained for the utility function. No two final solutions in the table are the same. However, the final utility value of most problems in the table are relatively close to one another. The major exception is problem twelve, which showed almost no improvement in the initial utility value, and which had a final solution 11.64 percent higher than the best final solution for the given utility function. The average error for the problems in the table is 3.46 percent.

TABLE 5.2
CONVERGENCE OF INTERACTIVE COMPUTER PROGRAM
USING DIFFERENT INITIAL POINTS

Problem Number	Number of Cities	Utility Function			Ideal Solution			Ideal Utility	Initial Point			Initial Utility	Final Solution			Final Utility	% Error
		α_1	α_2	α_3	TDIST	LDDEV	LNDEV		TDIST	LDDEV	LNDEV		TDIST	LDDEV	LNDEV		
1	32	0.291	0.026	0.683	809	130	9	275.68	809	1570	51	311.07	853	870	12	279.04	*
2	"	"	"	"	"	"	"	"	884	330	27	284.27	850	670	26	282.53	1.25
3	"	"	"	"	"	"	"	"	938	780	13	302.12	882	630	13	281.92	1.03
4	"	0.543	0.032	0.425	"	"	"	447.27	823	1500	43	513.16	828	420	47	483.02	*
5	"	"	"	"	"	"	"	"	875	400	22	497.28	834	600	51	493.74	2.22
6	"	"	"	"	"	"	"	"	824	2150	31	529.41	826	670	49	490.78	1.61
7	50	0.029	0.893	0.078	548	1	1	16.86	633	11	12	29.12	568	3	28	21.34	*
8	"	"	"	"	"	"	"	"	762	8	1	29.32	619	3	19	22.11	3.61
9	"	"	"	"	"	"	"	"	667	4	69	28.30	666	3	14	23.09	8.20
10	"	0.059	0.914	0.027	"	"	"	33.27	584	9	36	43.65	559	3	19	36.24	*
11	"	"	"	"	"	"	"	"	591	6	10	40.62	581	2	18	36.59	0.97
12	"	"	"	"	"	"	"	"	612	6	2	41.65	606	5	5	40.46	11.64
13	75	0.151	0.443	0.406	851	3	11	134.30	905	6	93	177.07	920	6	24	149.99	*
14	"	"	"	"	"	"	"	"	851	8	92	169.40	934	8	21	153.10	2.07
15	"	"	"	"	"	"	"	"	903	8	43	157.34	941	8	18	152.94	1.97

*Minimum value of utility function obtained from this starting point.

Nondominance Analysis. Another way of measuring the effectiveness of the procedure is to determine whether the final solution is a member of the nondominated set. Since each single-objective problem is solved heuristically, there is no guarantee that an optimal value for the single objective is reached. This carries over to the multiobjective case, in which there is no guarantee of nondominance.

One method of measuring nondominance is to consider all solutions which are obtained in the course of solving a WBVRP, comparing each solution against the others. As stated previously, any WBVRP can be expected to generate about six of these solutions. The first 200 such solutions obtained in the analyses of Chapter VI were taken as a data base for the nondominance study. Of these 200 solutions, 30 were found to be dominated. This translates into a 15 percent rate for dominated solutions. For most decision makers, this rate is probably acceptable. Of course, there is no way of knowing whether the dominating solutions were themselves dominated by other (undiscovered) solutions.

Efficiency of Program

The efficiency of the interactive program is measured by the time required to reach a final solution. The CPU time required for each of the single-objective functions was covered previously in Chapter IV. Although the interactive program includes a small amount of overhead for the main program and for the graphic display routines, this overhead is negligible. If the worst case for each single-objective algorithm for the problems of Chapter IV were experienced in solving a WBVRP, and if two of each of those single-objective problems were solved in the course of finding a final solution, then a total CPU time of over two minutes

could be expected. However, the majority of problems solved during this research required well under one minute total CPU time.

Another measure of efficiency is the total (clock) time for the analyst to arrive at the final answer to a WBVRP. This is a function of the amount of effort required to choose the least satisfactory achievement level, and to determine which tradeoff to select. For most of the problems in this research, the total clock time was less than ten minutes. Admittedly, had the author been solving real-world problems having more realistic tradeoff considerations, the total clock time might very well have been greater.

Example Problem

An example is now presented to demonstrate the use of the interactive computer program. The problem has 33 customers, a vehicle capacity of 150 units, and a distance limit of 50 miles per route. Distances are Euclidean. The problem details are given in Appendix B.

Figure 5.3 shows the initial route set presented to the decision maker. Recall that this route set is the best minimum distance solution chosen from eight successive runs of the Clarke and Wright savings algorithm, the 2-opt algorithm, and the 3-opt algorithm. This solution has a total distance of 174 miles, a route-load deviation of 27 units, and a route-length deviation of 22 miles. The decision maker is asked whether this solution is acceptable as a final solution. Since the driver of the longest route must travel more than twice the distance of the shortest route, the route set is not acceptable as a final solution. The decision maker selects route-length deviation as the objective to minimize.

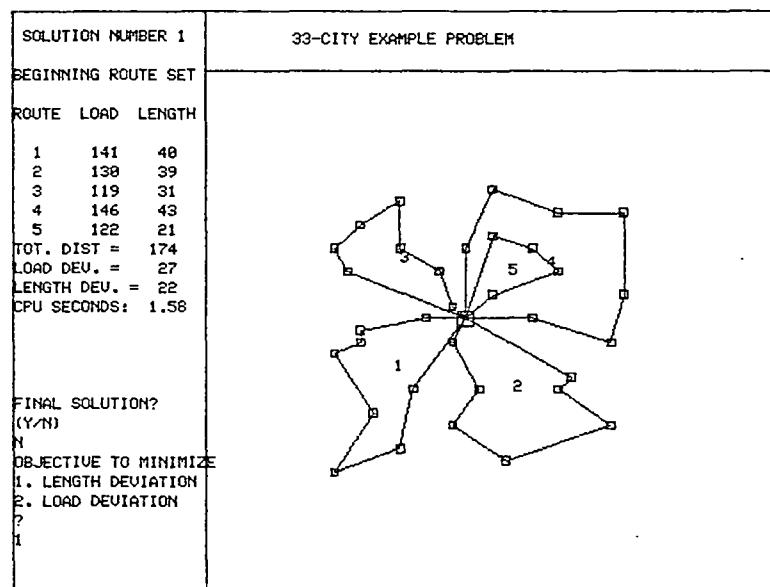
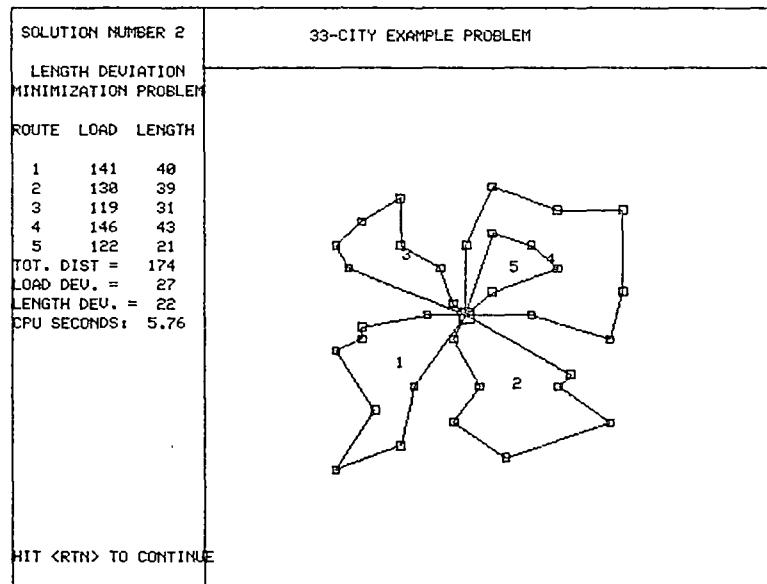


Figure 5.3. Example Problem: Beginning Route Set

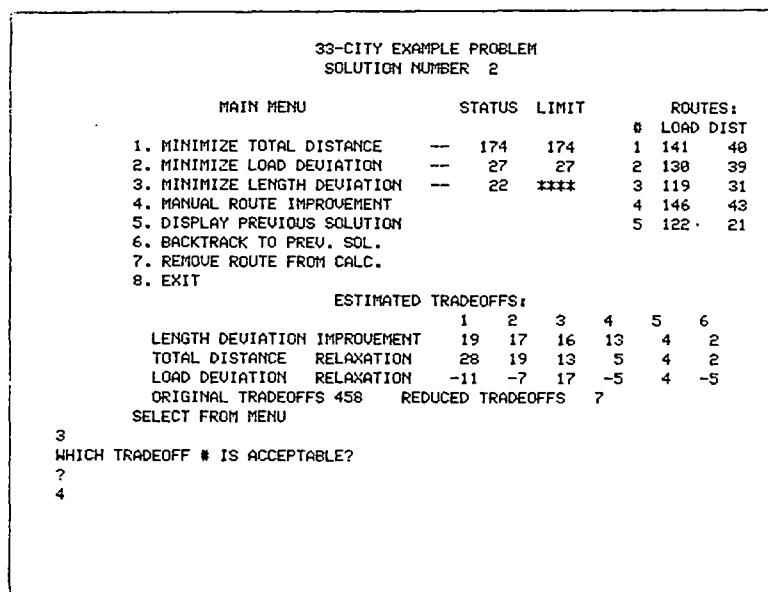
Figure 5.4(A) shows the results of the route-length deviation minimization problem (solution number 2). This problem was solved using the previous values of total distance (174 miles) and route-load deviation (27 units) as constraints, and was solved in an attempt to drive the solution to the nondominated set. The solution to this problem is the same as the previous one, which indicates that it probably lies in the nondominated set.

Figure 5.4(B) contains the menu, the problem status, and a set of tradeoffs resulting from solution number two. Since no progress was made in reducing route-length deviation, the achievement level of 22 miles is considered the least satisfactory achievement level, and the third menu item (minimize length deviation) is selected for the next iteration.

Since the objective function for the next iteration is the same as in the current iteration, some relaxation in at least one of the other two achievement levels is necessary. The tradeoff information shown in Figure 5.4(B) can be used to determine the amount of constraint relaxation to allow for total distance and route-load deviation. The fourth tradeoff indicates that an improvement of at least 13 miles in route-length deviation can be obtained if total distance is increased by 5 miles. This tradeoff also shows that the immediate result of the arc exchange associated with the tradeoff will be a decrease (indicated by the negative sign) of 5 units in route-load deviation. However, this negative value should not be used in determining which tradeoff to use, unless the competing tradeoffs are otherwise equivalent. The problem resulting from the tradeoff will actually have zero constraint relaxation for route-load deviation, as explained previously in 'Goal Tradeoffs'.



(A) ROUTE SET DISPLAY



(B) INTERACTIVE SCREEN DISPLAY

Figure 5.4. Example Problem: Solution Number 2

Having decided that an improvement of 13 miles in route-length deviation for an increase of 5 miles in total distance is an attractive tradeoff, the decision maker selects tradeoff number four, as shown in Figure 5.4(B). The problem to be solved in the next iteration is

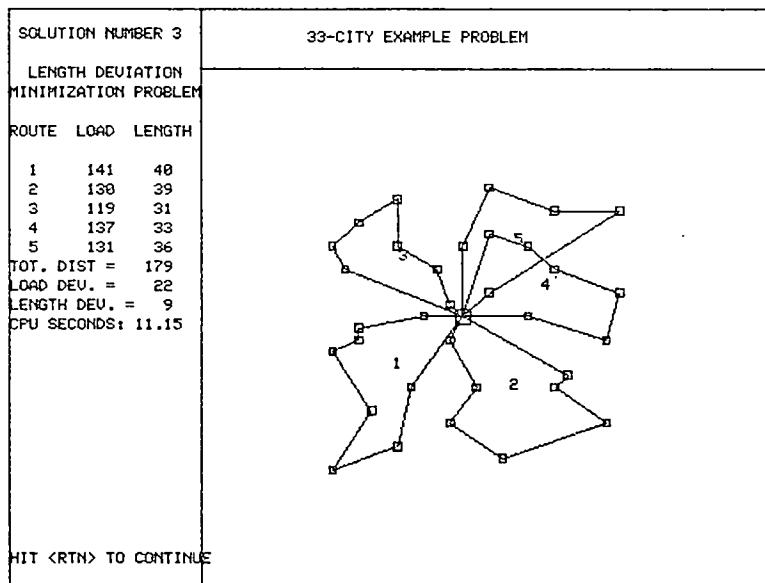
$$\text{Min LNDEV} \quad (5.10)$$

$$\text{S.T. TDIST} \leq 179 \quad (5.11)$$

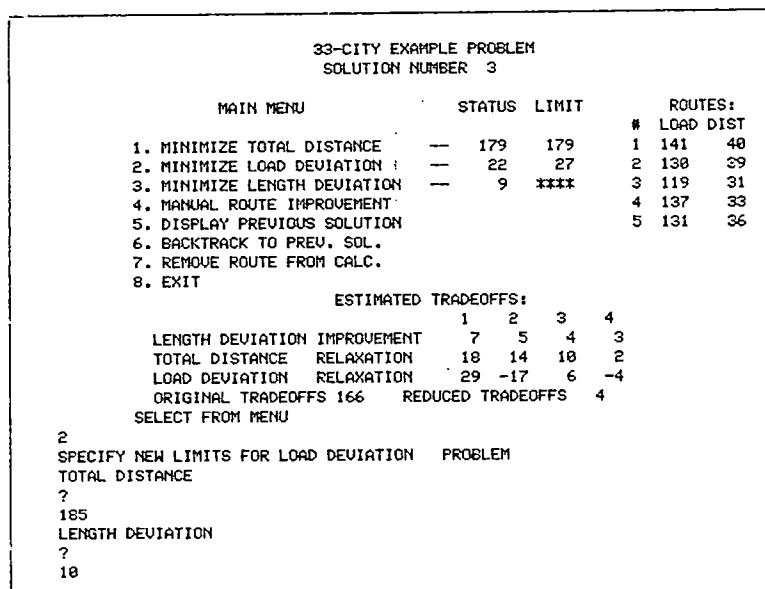
$$\text{and LDDEV} \leq 27. \quad (5.12)$$

Had the decision maker desired, a route-length deviation problem could have been solved without choosing one of the displayed tradeoffs. Entering a zero instead of a valid tradeoff number would result in a request for new constraints on total distance and route-load deviation. With these constraints, a route-length deviation problem would then be solved. This method generally yields inferior results to the tradeoff method, however, and can actually result in a dominated tradeoff which had been removed from the original tradeoff set. The use of one of the displayed tradeoffs guarantees at least the amount of improvement shown in the tradeoff, since the first thing the computer does is to perform the arc exchange associated with the tradeoff, before solving problem (5.10) - (5.12).

The solution to problem (5.10) - (5.12) is shown in Figure 5.5 (solution number 3). The route-length deviation algorithm has reduced the value of LNDEV to 9 miles, while TDIST has increased to 179 miles and LDDEV has increased to 27 units. This is the result tradeoff number four had indicated. After the tradeoff was made, no further improvement in the objective function could be made. The 'improvement ratio' for this tradeoff is unity.



(A) ROUTE SET DISPLAY



(B) INTERACTIVE SCREEN DISPLAY

Figure 5.5. Example Problem: Solution Number 3

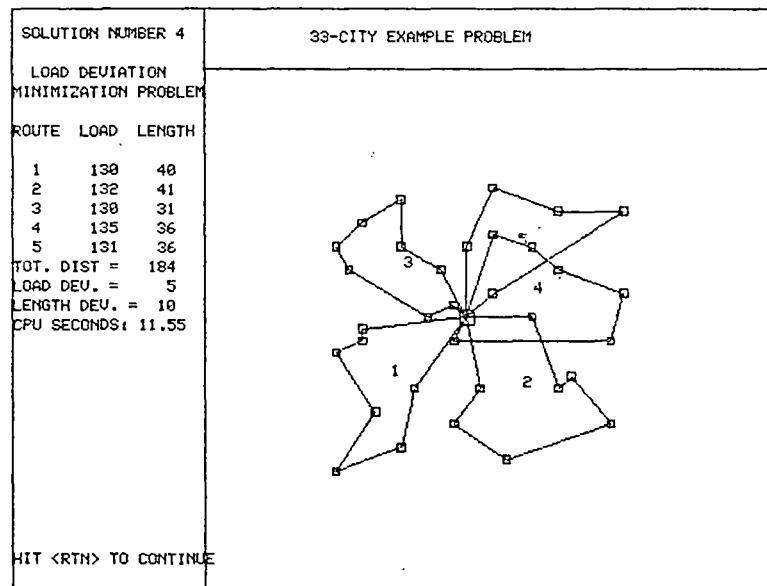
The decision maker must now choose the least satisfactory achievement level from the three values shown in Figure 5.5; i.e., total distance (179 miles), route-load deviation (22 units), or route-length deviation (9 miles). Suppose that route-load deviation is selected (menu option two). This response is shown in Figure 5.5(B). Since the new objective function is different from the previous one, the tradeoffs shown in Figure 5.5(B) will not be utilized. Instead, the decision maker is asked for new limits on the two new constraints, total distance and route-length deviation. The decision maker enters these limits as 185 miles and 10 miles, respectively. The new problem to be solved is

$$\text{Min } \text{LDDEV} \quad (5.13)$$

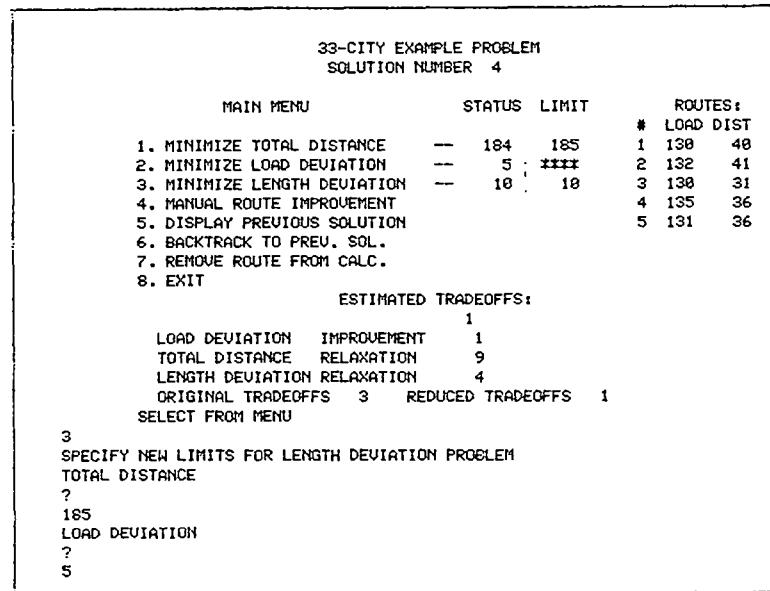
$$\text{S.T. } \text{TDIST} \leq 185 \quad (5.14)$$

$$\text{and } \text{LNDEV} \leq 10. \quad (5.15)$$

Figure 5.6 shows the solution to problem (5.13) -(5.15). Route-load deviation has been reduced to 5 units, while total distance is now 184 miles and route-length deviation is 10 miles. Once again, the decision maker must decide which of the three achievement levels is least satisfactory. Suppose that route-length deviation is selected. The objective function has changed, so the single tradeoff displayed in Figure 5.6(B) is not used. Instead the decision maker is asked for new limits on the two constraints, total distance and route-load deviation. At this point, the decision maker is willing to relax the constraints very little. The new limits are 185 miles and 5 units, respectively. The new problem to be solved is:



(A) ROUTE SET DISPLAY



(B) INTERACTIVE SCREEN DISPLAY

Figure 5.6. Example Problem: Solution Number 4

$$\text{Min LNDEV} \quad (5.16)$$

$$\text{S.T. TDIST} \leq 185 \quad (5.17)$$

$$\text{and LDDEV} \leq 5. \quad (5.18)$$

The solution to problem (5.16) - (5.18) is shown in Figure 5.7.

The route-length deviation has not been decreased, due to the very slight relaxation (one mile) the decision maker allowed in the total distance constraint. However, the decision maker is now presented with a set of tradeoffs (Figure 5.7(B)) which will allow a better evaluation of the effects of constraint relaxation. In fact, allowing no constraint relaxation at all and solving a problem just to obtain such a set of tradeoffs is an acceptable practice using the interactive program. The second tradeoff indicates that route-length deviation can be reduced by three miles if total distance is increased by two miles and route-load deviation is increased by four units. The decision maker selects this tradeoff, as indicated by the response in Figure 5.7(B).

The new problem to be solved is

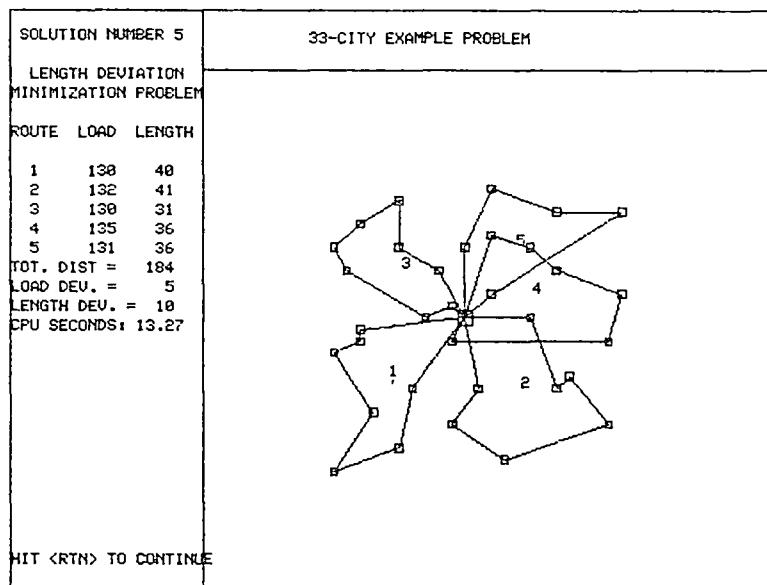
$$\text{Min LNDEV} \quad (5.19)$$

$$\text{S.T. TDIST} \leq 186 \quad (5.20)$$

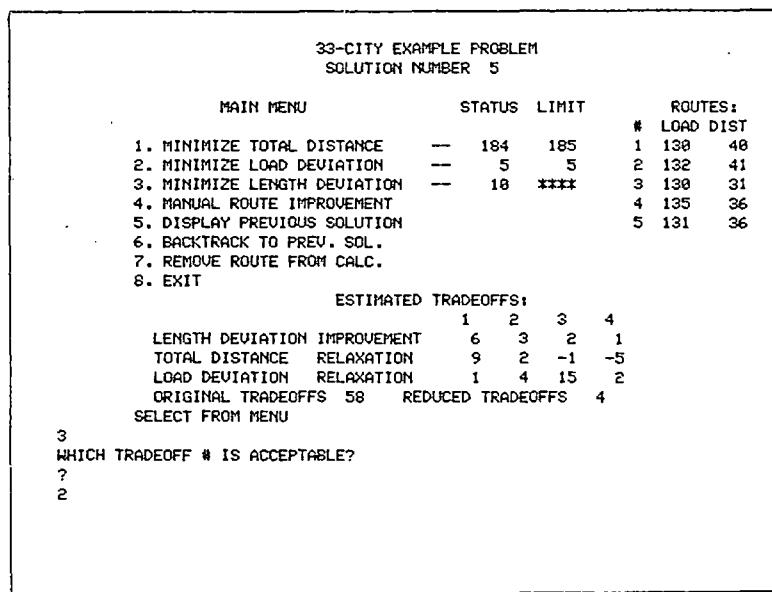
$$\text{and LDDEV} \leq 9. \quad (5.21)$$

The solution to problem (5.19) - (5.21) is shown in Figure 5.8.

The total distance of the set of routes shown in Figure 5.8(A) is 186 miles, route-load deviation is 9 units, and route-length deviation is 7 miles. At this point, the decision maker is unable to select one of these achievement levels as least satisfactory, so the procedure is halted by selecting menu option number eight. Comparing this final solution to the initial (minimum distance) solution, it is seen that

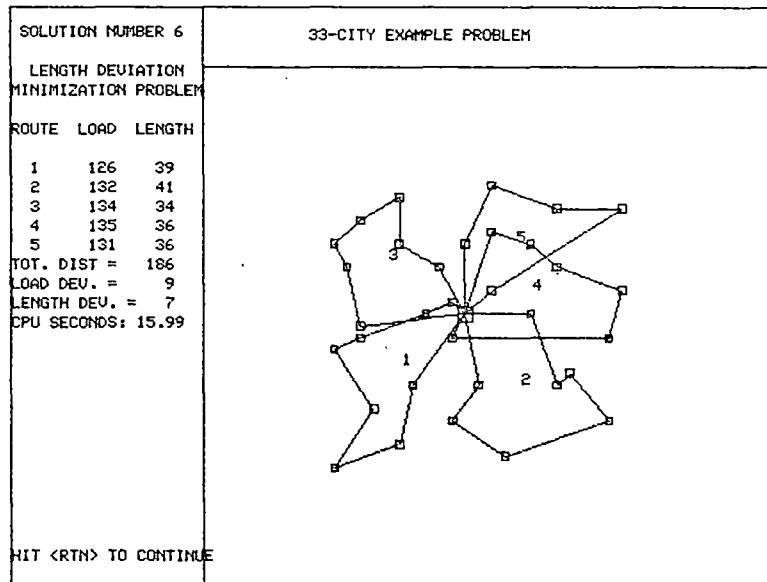


(A) ROUTE SET DISPLAY

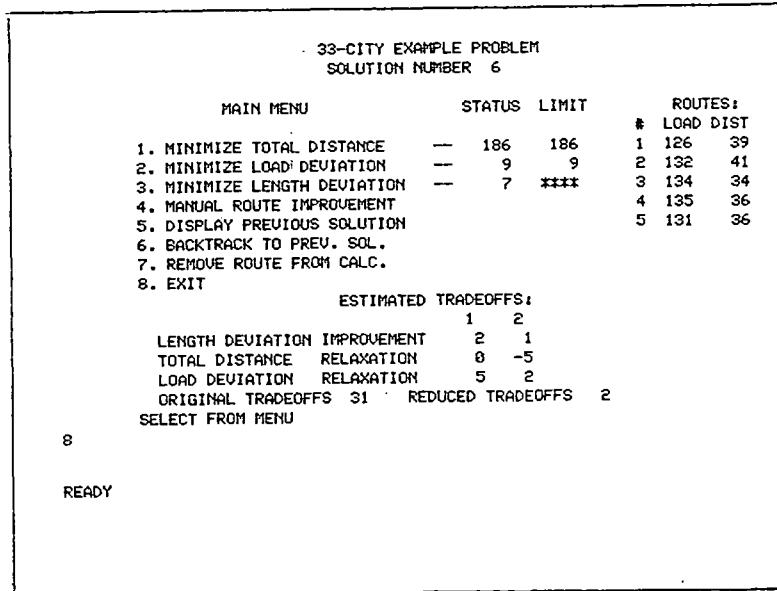


(B) INTERACTIVE SCREEN DISPLAY

Figure 5.7. Example Problem: Solution Number 5



(A) ROUTE SET DISPLAY



(B) INTERACTIVE SCREEN DISPLAY

Figure 5.8. Example Problem: Solution Number 6

route-load deviation has been reduced by 66.67 percent and route-length deviation has been reduced by 68.18 percent. These improvements in workload balance were made at a penalty of 6.90 percent in total distance.

Summary

In this chapter, the single-objective algorithms of Chapter IV have been consolidated into an interactive computer program to enable the decision maker to solve the WBVRP using a heuristic version of the Method of Satisfactory Goals. Because dual variables are not available, the program provides the user with tradeoffs which determine the amount of constraint relaxation at a given iteration of the procedure. The tradeoffs were evaluated in terms of their ability to predict improvement in an objective function using a specified level of constraint relaxation. The effectiveness of the interactive program was evaluated in terms of its ability to converge to a solution from different starting points, and by the percentage of dominated solutions generated by the procedure. The efficiency of the program was evaluated in terms of the amount of time (both CPU and elapsed) required to reach a final solution. Finally, the use of the interactive computer program was demonstrated by solving a sample WBVRP.

CHAPTER VI

WORKLOAD BALANCING COSTS

Introduction

A set of workload balanced routes can be expected in most cases to be more costly in terms of the total distance driven. This extra distance will translate to added fuel and maintenance costs for the fleet and, depending on the particular distribution system, might translate to added driver and/or helper costs. In the sample problem of Chapter V, the extra distance was 6.90 percent. Other costs, such as administration, dispatching, hardware and software, might or might not increase. These other costs are not addressed in this chapter.

The cost penalty for workload balancing will depend on the extent to which the routes are to be balanced. In addition, the depot location, customer demand pattern, and spatial characteristics (customer location pattern) of the problem can be expected to affect the penalty. In this chapter, it is assumed that routes are to be balanced as much as possible, allowing the analysis to concentrate on the effects of depot location and demand and spatial characteristics.

Attention is focused on distribution systems having unbalanced workloads. It is desired to know what penalty must be paid in going from unbalanced, minimum distance routes to routes which are balanced in one or both of the workload elements, and the effect that the problem characteristics have on the penalty. The workload balancing penalty is

defined as the fraction of total route distance which must be increased in order to balance the workload element(s).

It is easy to imagine situations in which route balancing is affected by having some degree of variability in the workload element being balanced. Consider a route-load balancing problem in which demand is constant and two route loads differ by a given amount, say, twice the constant demand. Since all demands are the same, no simple pairwise exchange can balance the loads. However, if the demand varies from customer to customer, there is a chance that fewer exchanges will balance the loads and that the resulting distance penalty will be less. Of course, depending on circumstances, the opposite effect could result, requiring more exchanges and a greater distance penalty. And, if only one workload element is being balanced, the degree of variability in the workload element not being balanced can affect the total distance penalty. In the present example, variation in the distances between customers could result in the total distance penalty being greater or less than would result under a uniform spatial pattern.

The purpose of the analysis, then, is to determine whether any conclusions can be made regarding the effect that a problem's characteristics have on the workload balancing penalty. The analysis is performed for route load balancing, route length balancing, and balancing of both route loads and route lengths.

Method of Analysis

Two different approaches could be taken in analyzing the effects of depot location, customer demand pattern, and customer spatial pattern on the workload balancing penalty. In the first approach, a large random

sample of problems having different characteristics could be solved, followed by a statistical analysis of the results. In the second, a 'standard' problem could be set up, then solved several times as different characteristics are systematically varied while holding other characteristics constant. The second approach was taken in this analysis, primarily because it was felt that more insight could be gained by thoroughly studying the standard problem, but also because of the amount of effort which would be involved in creating and checking out each new randomly generated problem before solving it. It is realized that any conclusions reached by this type of analysis will not necessarily apply to all problems. However, the results might lead to hypotheses which can be tested through further research.

The standard problem used in the analysis contains 36 customers served by four vehicles having a capacity of 110 units each and no limit on route length. The customers have an average demand of approximately ten units each. Distances are Euclidean. To study the effect of demand variability, four different demand patterns are used. The first pattern has a constant demand of ten units with no variability. The second pattern has an average demand of 9.78 and a standard deviation of 1.51, and was created by randomly generating integer values between 8 and 12, inclusive, from a uniform distribution. The third pattern has an average demand of 9.78 and a standard deviation of 2.12, created by randomly generating values between 7 and 13, inclusive, from a uniform distribution. Finally, a fourth demand pattern has an average of 9.83 and a standard deviation of 3.73, having integer values between 4 and 16, inclusive. Each demand in the four patterns is shown in Table 6.1.

To study the effect of spatial characteristics, a six-by-six grid

TABLE 6.1
CUSTOMER DEMAND PATTERNS

Customer	Demand Pattern 1	Demand Pattern 2	Demand Pattern 3	Demand Pattern 4
1	10	8	13	4
2	10	9	9	10
3	10	10	10	15
4	10	12	9	14
5	10	8	11	5
6	10	10	11	15
7	10	8	7	6
8	10	9	13	13
9	10	9	7	13
10	10	12	12	13
11	10	11	8	10
12	10	8	10	5
13	10	11	10	8
14	10	8	7	16
15	10	12	8	6
16	10	11	11	5
17	10	9	7	12
18	10	8	13	11
19	10	11	8	10
20	10	8	11	7
21	10	12	8	12
22	10	9	7	13
23	10	8	12	8
24	10	12	12	13
25	10	11	10	9
26	10	8	11	5
27	10	8	8	8
28	10	9	13	12
29	10	12	8	12
30	10	10	7	14
31	10	12	12	12
32	10	10	9	4
33	10	10	12	4
34	10	8	7	6
35	10	10	8	16
36	10	11	13	8

pattern was established and customers were located on the grid. The grid points are separated by ten distance units in the vertical and horizontal directions. Four different spatial patterns were then established. The first spatial pattern has each of the 36 customers located on one of the grid points. The second pattern has each customer randomly placed within plus or minus three distance units in the vertical and horizontal directions from a grid point. The third and fourth patterns have customers randomly placed within plus or minus five distance units and within plus or minus ten distance units from the grid points, respectively. The four different customer location patterns are shown in Figures 6.1 through 6.4. An examination of these figures shows an increase in the variability of distances between customers with the first through the fourth pattern, respectively. One measure which could be used to quantify the spatial characteristic is the standard deviation of distances between nearest neighbors divided by the average distance between nearest neighbors. This measure is found to be 0.00, 0.16, 0.38, and 0.42 for the first through fourth customer location pattern, respectively. This measure of spatial dispersion may or may not be useful in predicting the workload balancing penalties. In either case, spatial pattern one is referred to as 'highly structured', pattern two is referred to as 'somewhat structured', and patterns three and four are referred to as 'unstructured'.

To study the effect of depot location on the workload balancing penalty, the depot was moved from the centroid to the outer bound of the location pattern in different problem steps. Then for each combination of demand pattern, spatial pattern, and depot location, four different problems were solved. The objective functions to be minimized in the

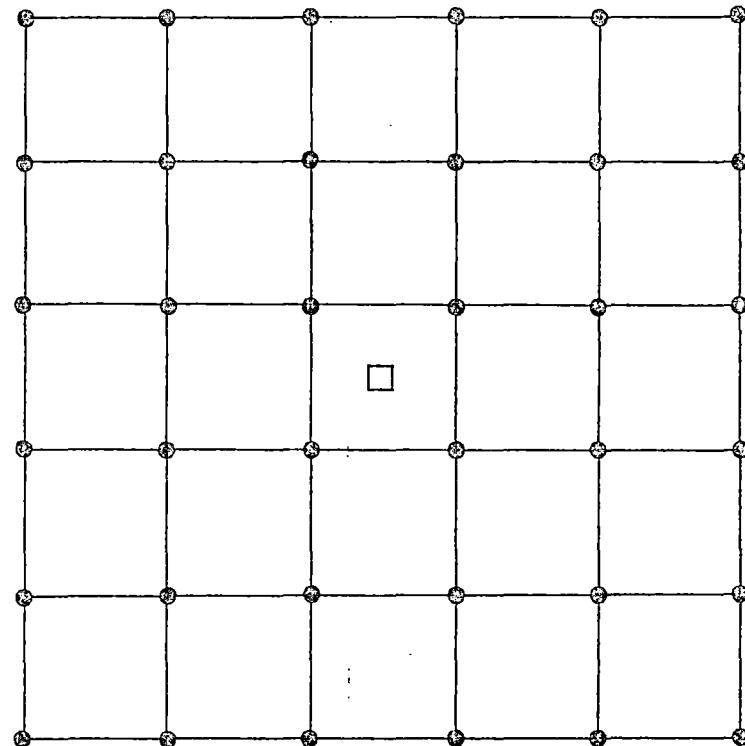


Figure 6.1. First Customer Location Pattern

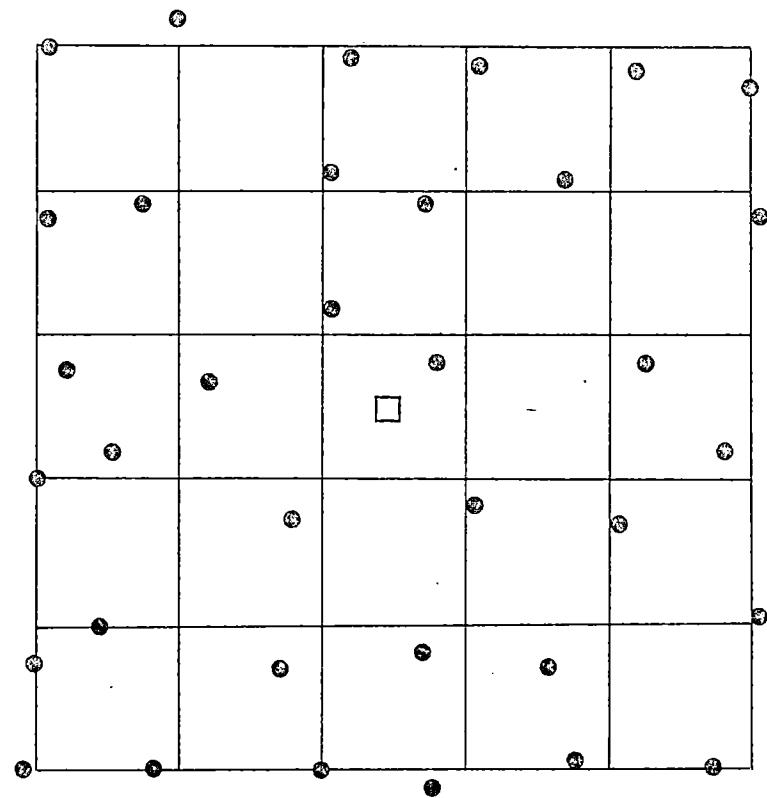


Figure 6.2. Second Customer Location Pattern

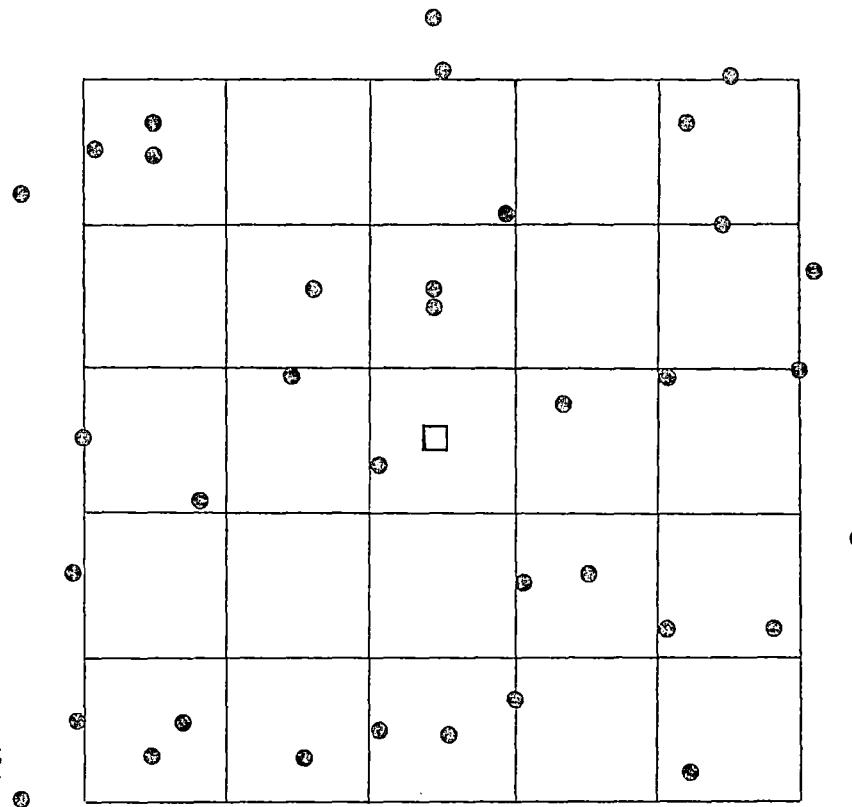


Figure 6.3. Third Customer Location Pattern

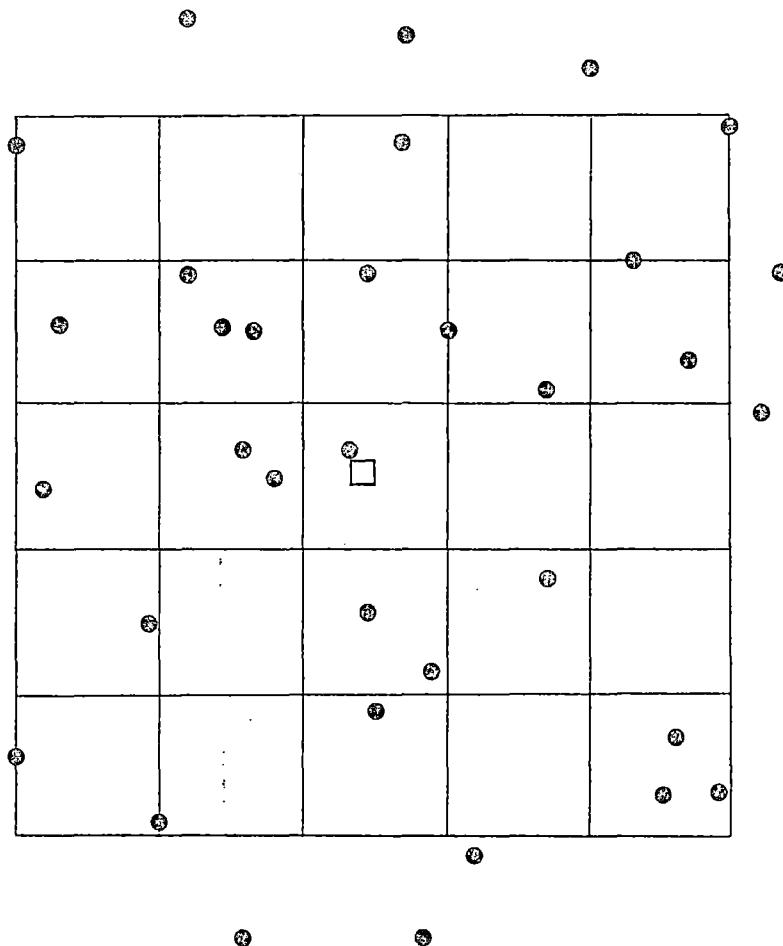


Figure 6.4. Fourth Customer Location Pattern

four problems were:

1. Total distance
2. Route-load deviation
3. Route-length deviation
4. Route-load and route-length deviation, equally weighted.

A total of 192 WBVRPs were solved using the interactive program of Chapter V.

Results

Depot Location Penalty

Before addressing the problem of workload balancing penalties, it is of interest to know the effect that depot location has on the costs of a distribution system that operates with minimum total distance routes. If the total distance of the routes increases as the depot is located away from the centroid of customer locations, then a 'depot location penalty' is paid by that distribution system. For a particular combination of demand pattern, spatial pattern, and depot location, the depot location penalty is calculated as

$$P_{Loc} = (TDIST_{Min} - TDISTC_{Min})/TDISTC_{Min} \quad (6.12)$$

where P_{Loc} = depot location penalty,

$TDIST_{Min}$ = total distance obtained in distance minimization problem at the given depot location,

and $TDISTC_{Min}$ = total distance obtained in distance minimization problem at the centroid.

The depot location penalties are plotted in Figure 6.5 as a function of the depot distance from the centroid (expressed as a fraction of the distance from the centroid to the grid boundary), demand pattern, and

spatial pattern. The penalty is seen to increase as the depot moves away from the centroid. This can be explained by the fact that routes formed when the depot is located at the centroid tend to be non-overlapping; but as the depot is moved away from the centroid, the routes must intersect in order to maintain vehicle capacity constraints. This intersecting of the routes causes the total distance, and thus the penalty, to be increased. The lowest penalties are obtained when demand is constant and the spatial pattern is highly structured (pattern number one). Any deviation from this combination causes the penalties to worsen.

For the most part, the depot location penalties are related to the spatial patterns, but not as a function of the simple measure of spatial dispersion defined above. In fact, the penalties for spatial patterns two and four are similar, yet the two patterns are totally different, pattern two being somewhat structured and pattern four being unstructured. This indicates that the simple measure of spatial dispersion is inadequate to predict the depot location penalties.

The effect of demand variability on the depot location penalty can be seen in Figure 6.5. For a very structured spatial pattern, the penalty is greatly increased as demand variability is increased. For the other spatial patterns, the penalty is decreased for some demand patterns, increased for others. The overall effect of increased variability of demand is to decrease the spread between the highest and lowest penalty values, lessening the effect of spatial pattern. It is likely that the effect of spatial pattern would be shown to be even less if demand variability were increased beyond that of demand pattern four, eventually taking on values close to those shown for the highly structured spatial pattern (number one) in Figure 6.5.

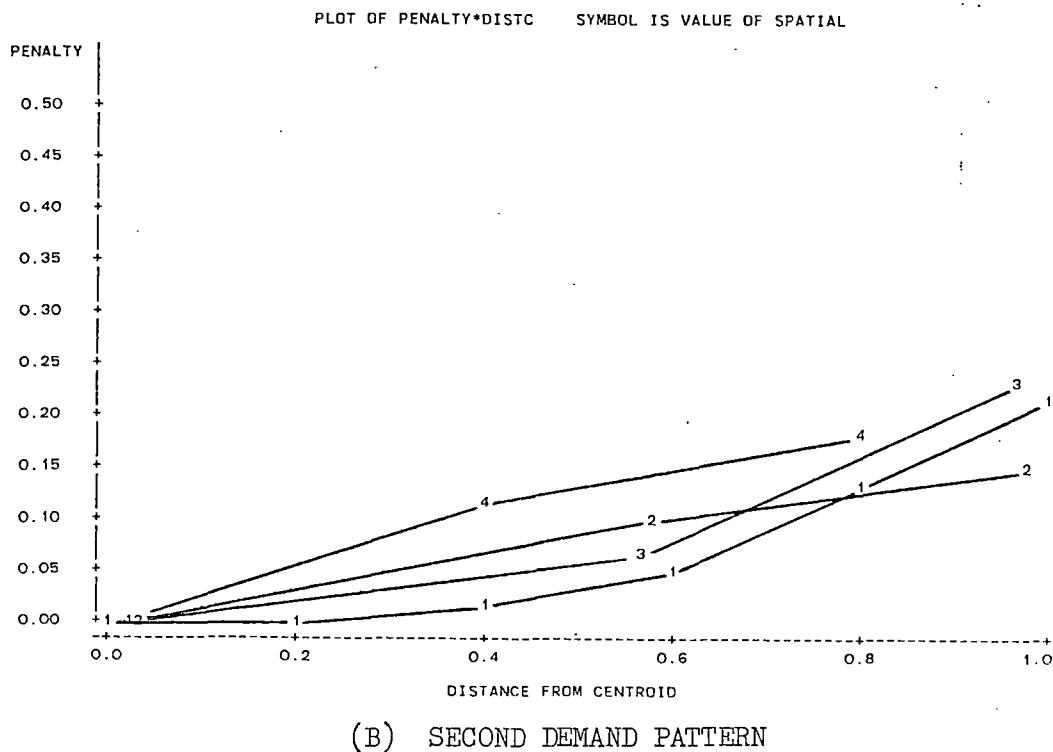
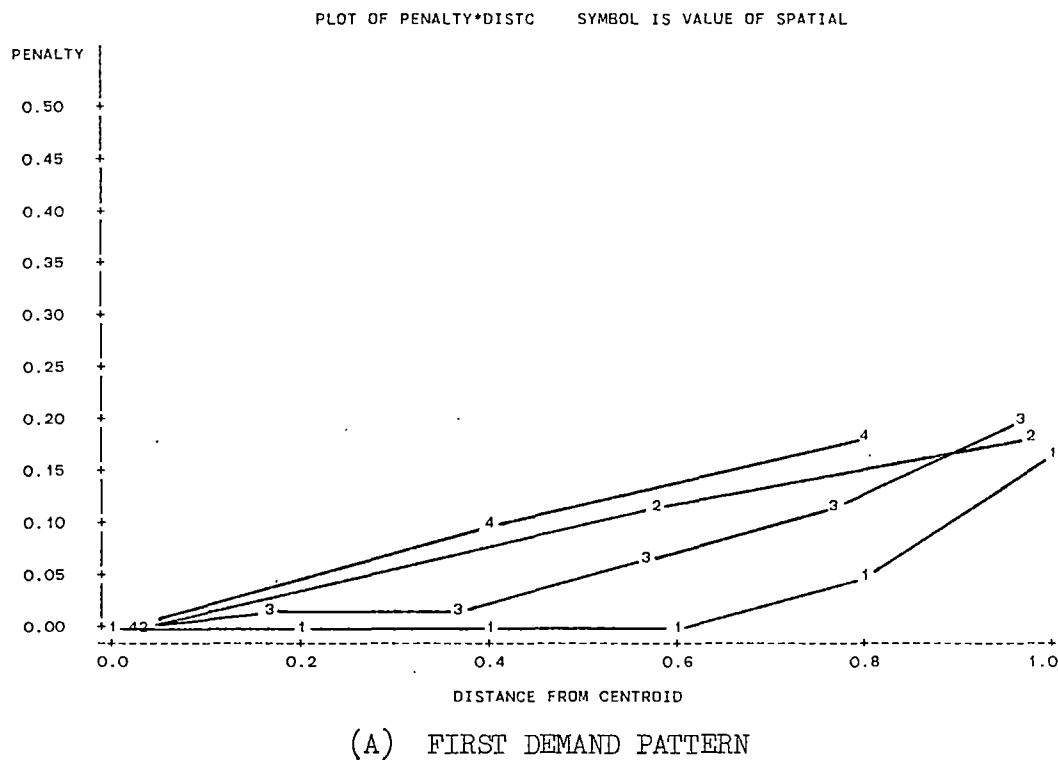


Figure 6.5. Depot Location Penalties for Minimum-Distance Problems

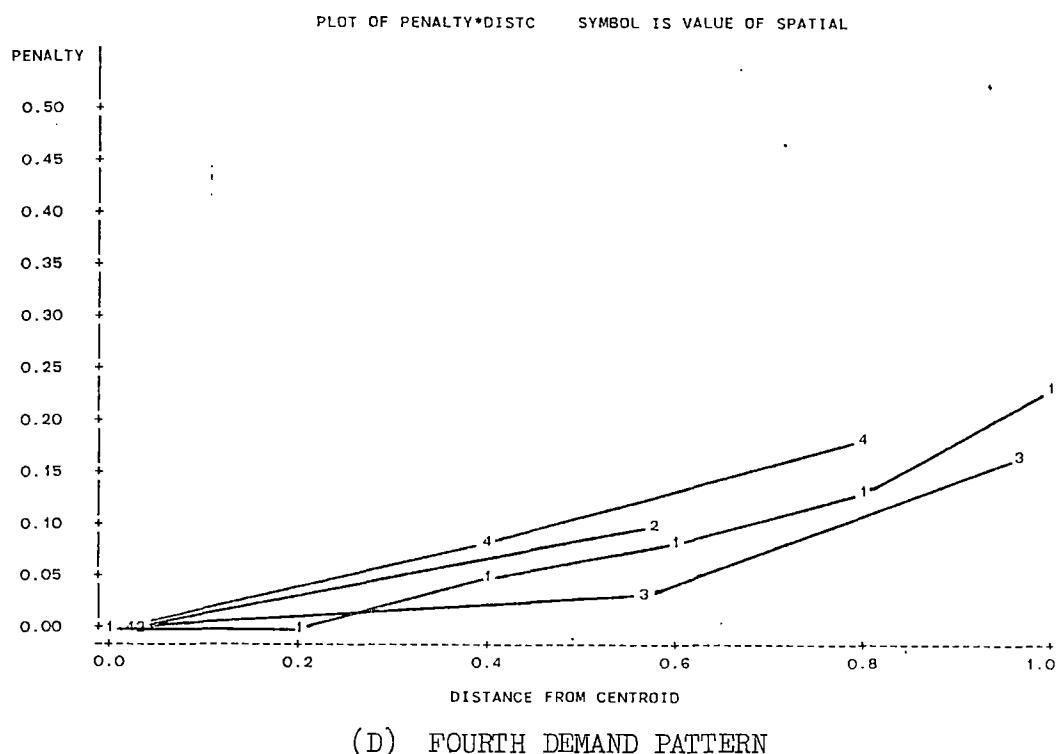
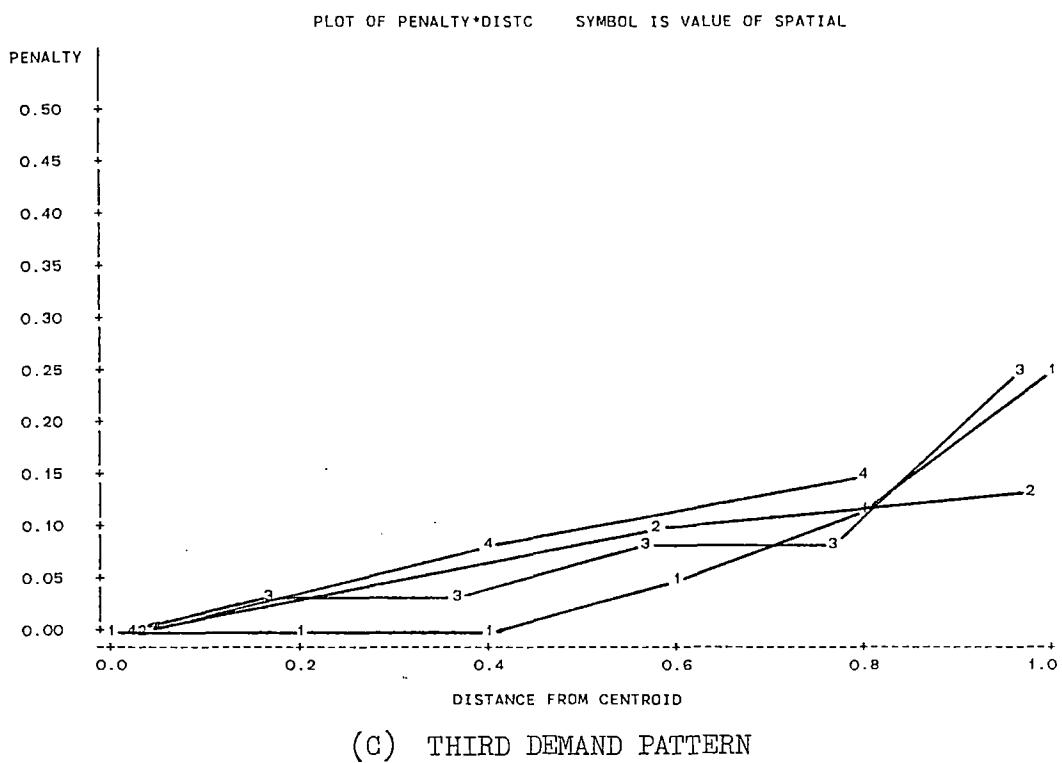


Figure 6.5. (Continued)

To be competitive, the distribution system should have its depot as close to the centroid as possible. The depot location penalties for minimum distance routes can be thought of as 'base case' penalties, to which are added the workload balancing penalties of the next section. For instance, if a 0.10 depot location penalty is considered the highest that could be tolerated, then the depot could be located as far away as 80 percent of the distance to the grid boundary or no farther away than 40 percent, depending on the particular combination of demand and spatial patterns. However, if the depot is located at the 0.10 penalty limit, workload balancing would not be an attractive option because the distribution system could not pay the added penalty for workload balancing and remain competitive. For this reason, in the next section it will be assumed that the depot is not located farther away than 60 percent of the distance, an average of the two extremes.

Workload Balancing Penalties

For a particular combination of demand pattern, spatial pattern, and depot location, the workload balancing penalties are calculated as follows:

$$P_{LDDEV} = (TDIST_{LDDEV} - TDIST_{Min})/TDIST_{Min}, \quad (6.2)$$

$$P_{LNDEV} = (TDIST_{LNDEV} - TDIST_{Min})/TDIST_{Min}, \quad (6.3)$$

$$\text{and } P_{BOTH} = (TDIST_{BOTH} - TDIST_{Min})/TDIST_{Min}, \quad (6.4)$$

where P_{LDDEV} = penalty for route-load balancing,

P_{LNDEV} = penalty for route-length balancing,

P_{BOTH} = penalty for balancing both route load and route length,

$TDIST_{LDDEV}$ = total distance obtained in route-load deviation problem,

$TDIST_{LNDEV}$ = total distance obtained in route-length deviation problem,

$TDIST_{BOTH}$ = total distance obtained in route-load and route-length deviation problem,

and $TDIST_{Min}$ = total distance obtained in distance minimization problem.

Route-load Balancing. The route-load balancing penalties are plotted in Figure 6.6 as a function of depot distance from the centroid, demand pattern, and spatial pattern. In Figure 6.6(A) it can be seen that, for the system with constant demand, route-load balancing is an attractive option regardless of the spatial pattern if distribution management is willing to pay a penalty of say, 0.10 over the minimum distance routing costs. With any demand variability at all, however, the penalty is a complex function of the distance from the centroid, demand variability, and spatial pattern. Only for a highly structured spatial pattern (number one) can the penalty be expected to be acceptable regardless of demand, and this spatial pattern is highly unlikely in real-world problems.

Route-length Balancing. The route-length balancing penalties are plotted in Figure 6.7 as a function of depot distance from the centroid, demand pattern, and spatial pattern. Here it can be seen that the penalty tends to be less for the structured location patterns (one and two) than for the unstructured patterns (three and four). Using a penalty limit of 0.10, the unstructured location patterns would not yield acceptable penalties except in a few cases, and then only if the depot were located near the centroid. The effect of demand variability is not apparent.

Route-load and Route-length Balancing. The penalties for balancing both route loads and route lengths are plotted in Figure 6.8. The plots are very similar to those of Figure 6.7, and the same general comments apply. Overall, the penalties are slightly higher than those of Figure 6.7, as could be expected. In some cases the penalty is lower, but this is because solutions were accepted in those cases which did not obtain the absolute minimum of both route balancing measures, but rather a very low value of each. The implication is that in those situations in which route lengths are balanced, the route loads could also be balanced if distribution management is willing to make minor tradeoffs between the two.

Conclusions

The results for the 36-city 'standard' problem were not as conclusive as had been hoped. In many cases the costs of workload balancing, and therefore the economic attractiveness of using workload-balanced routes, depend on a complex interaction of the problem variables. However, the following statements can be made:

1. For minimum-distance routes, the total route distance is least if the depot is situated near the center of all customer locations. Distribution systems having such centrally located depots are therefore more able to pay an additional cost for workload balancing than are those distribution systems not having centrally located depots.
2. If demand is constant, balancing of route loads causes a relatively small penalty regardless of the spatial pattern of the customers. This should have implications not only for those

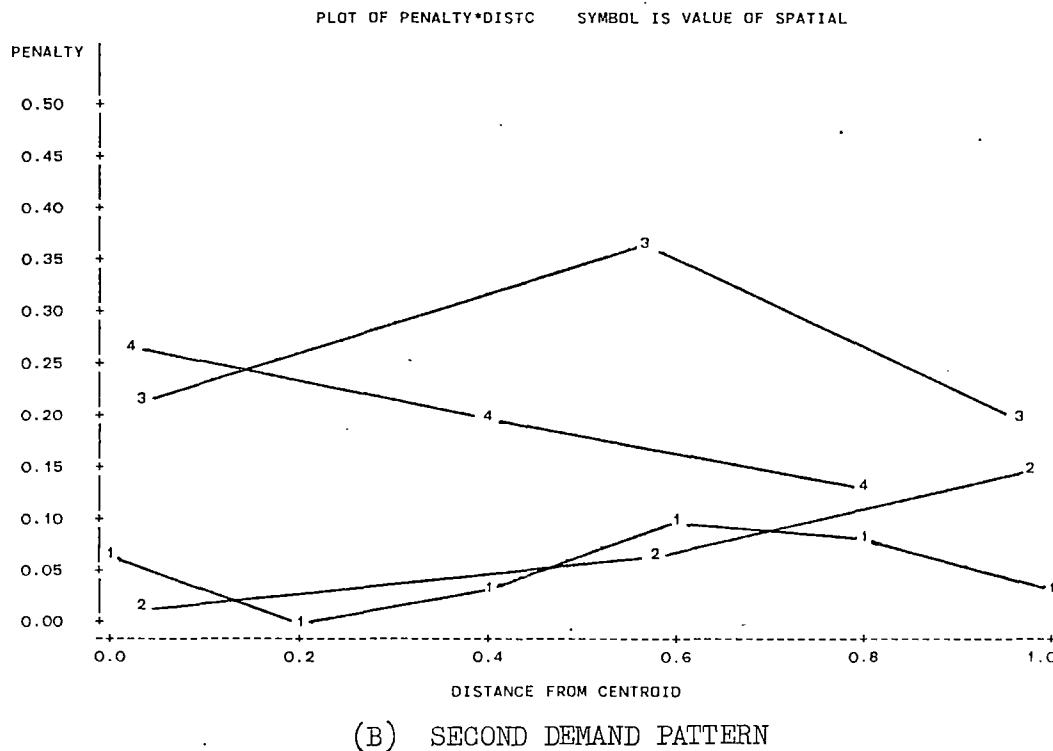
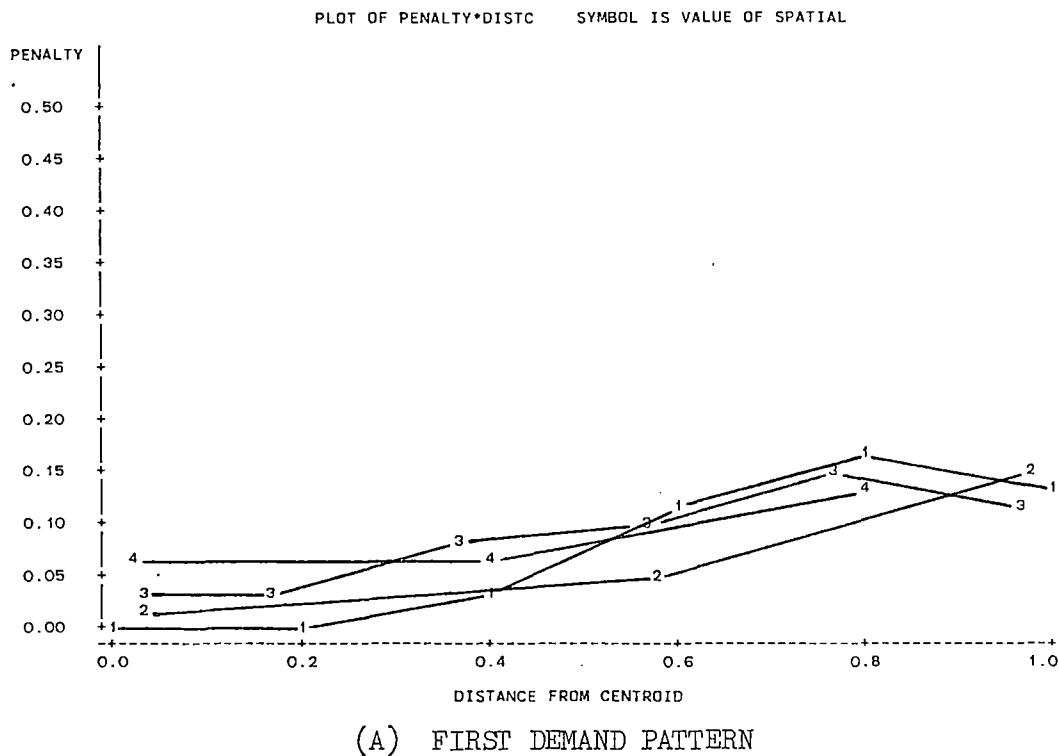


Figure 6.6. Route-Load Balancing Penalties

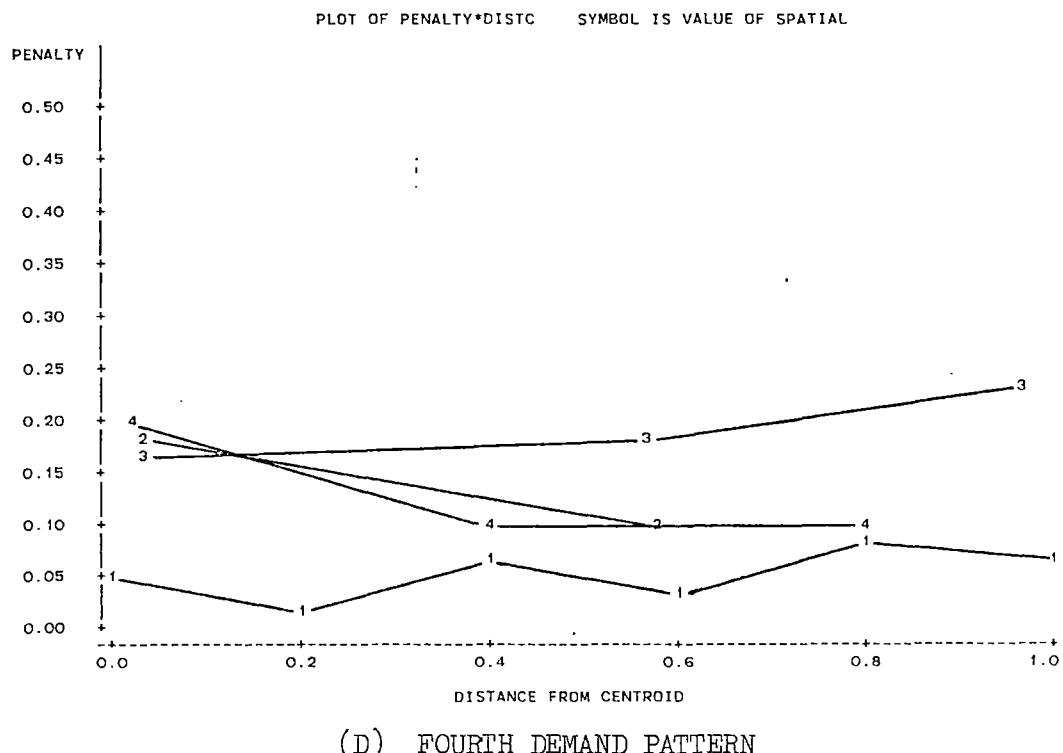
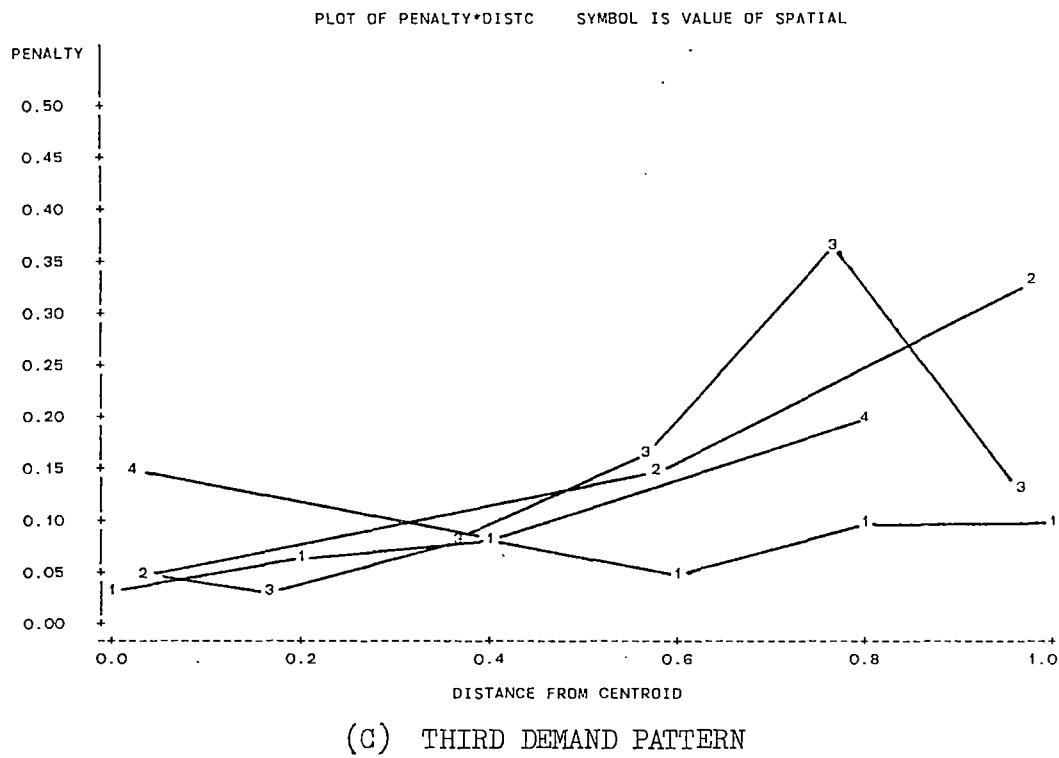


Figure 6.6. (Continued)

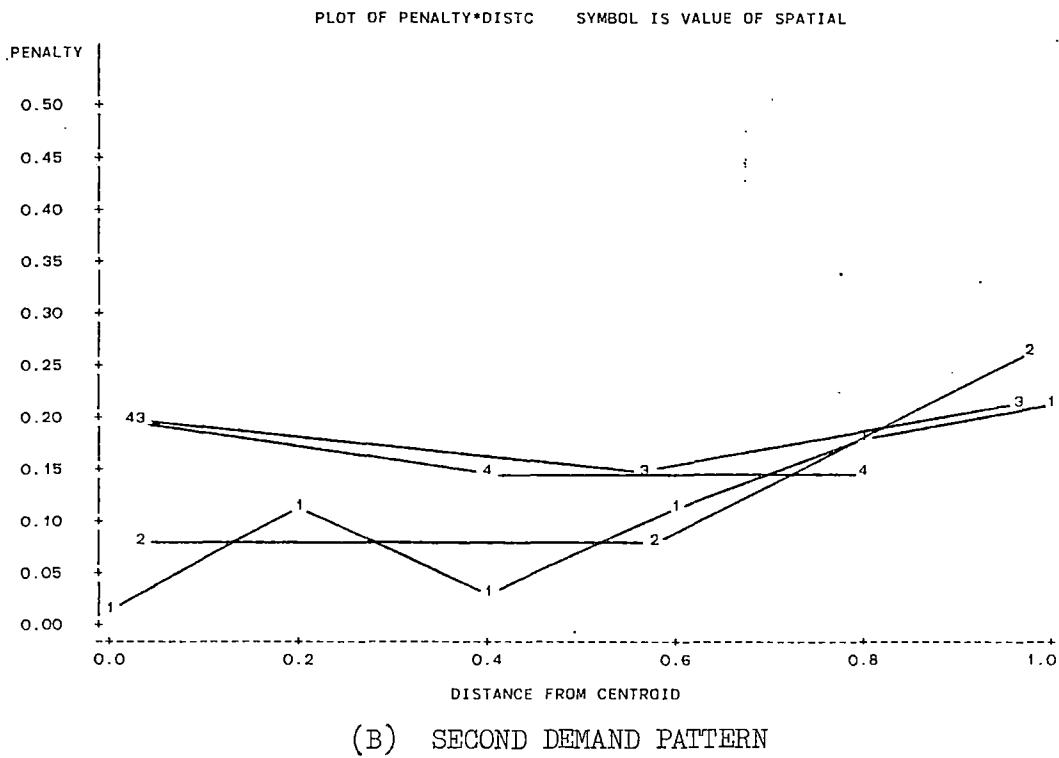
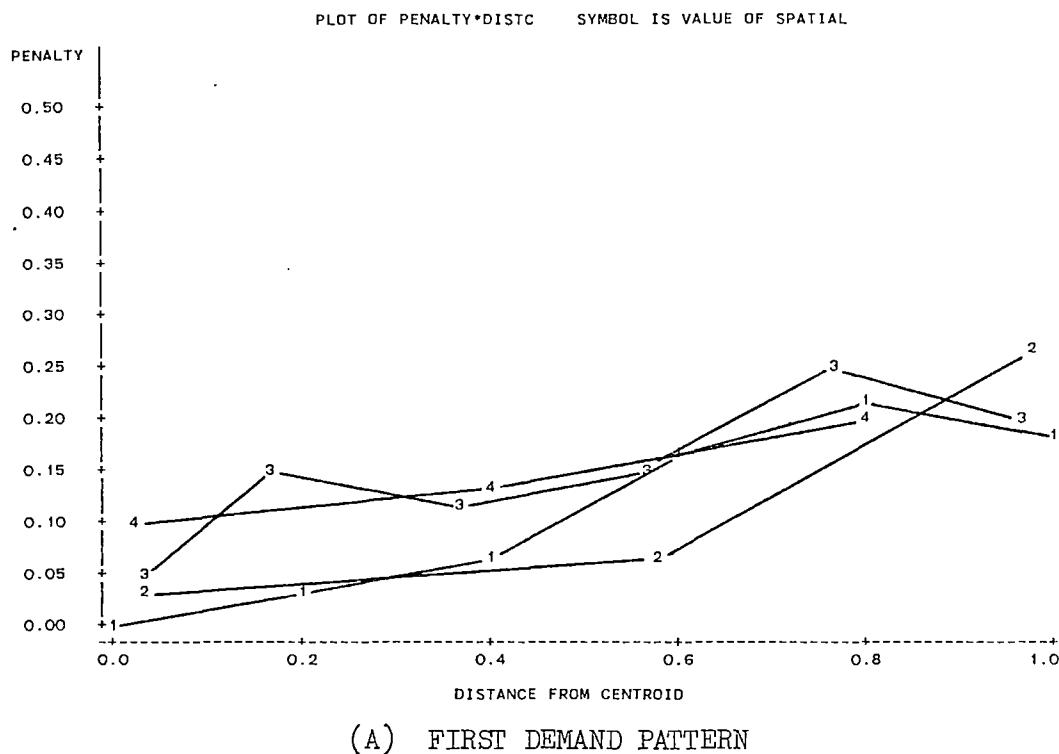


Figure 6.7. Route-Length Balancing Penalties

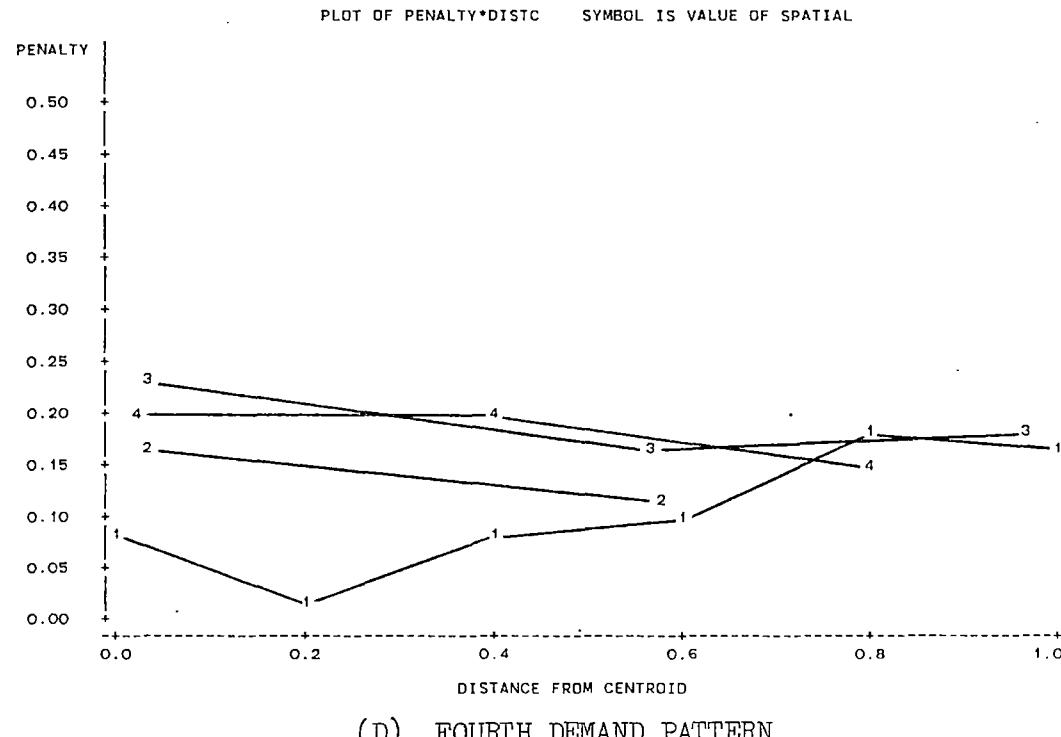
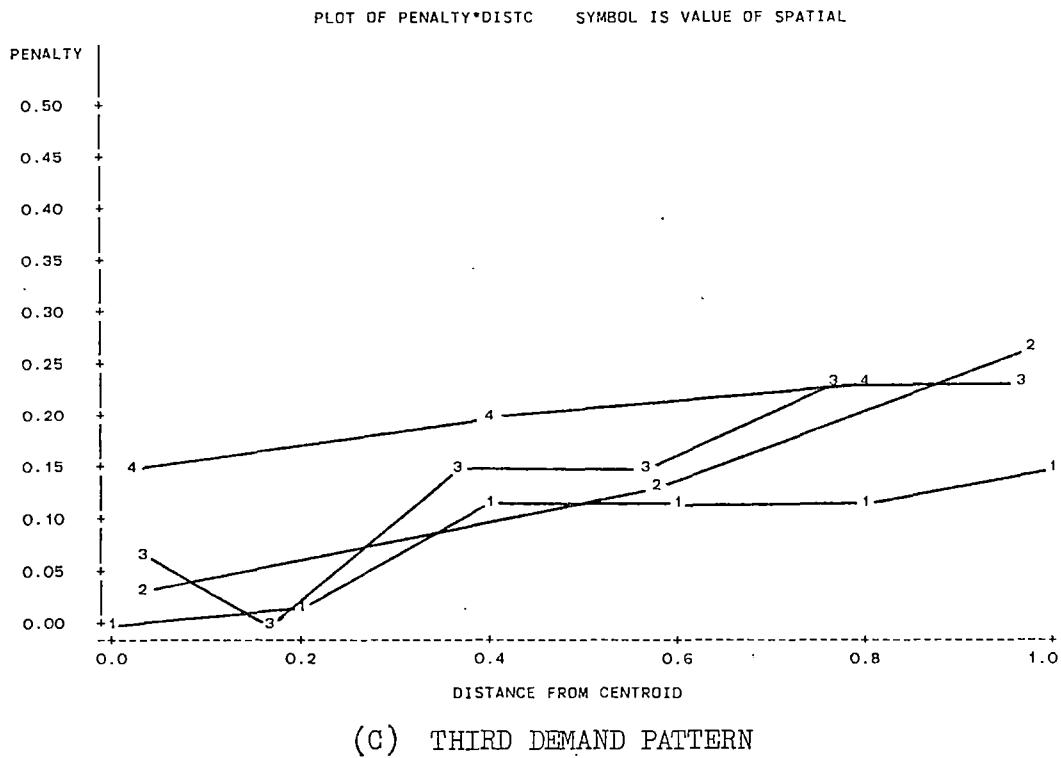


Figure 6.7. (Continued)

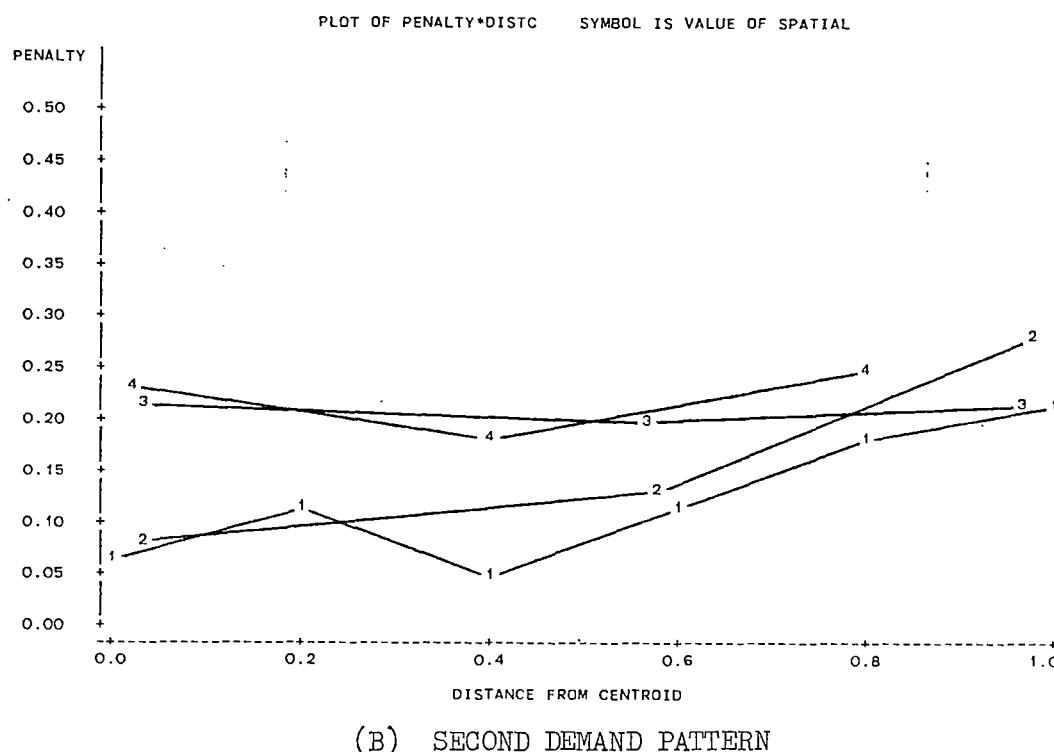
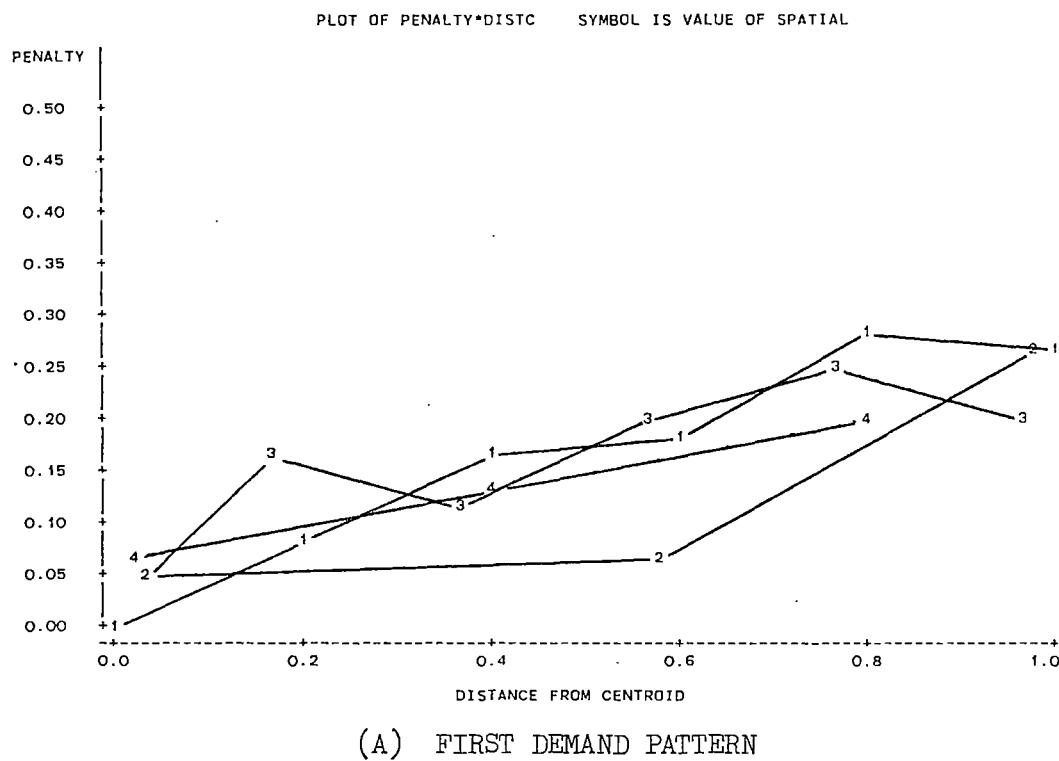
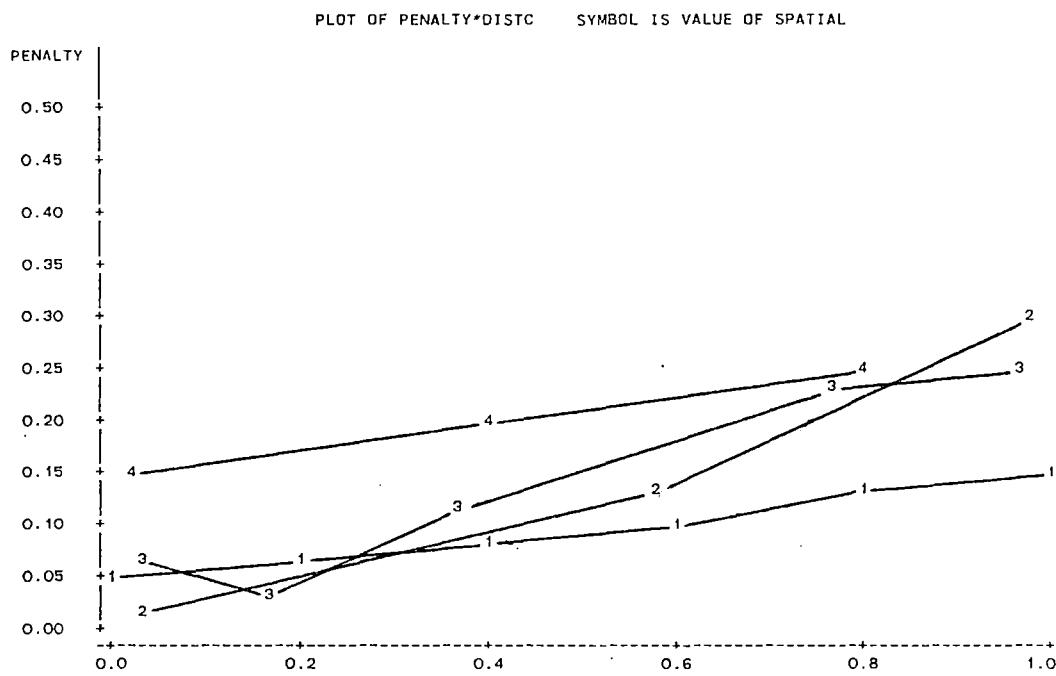
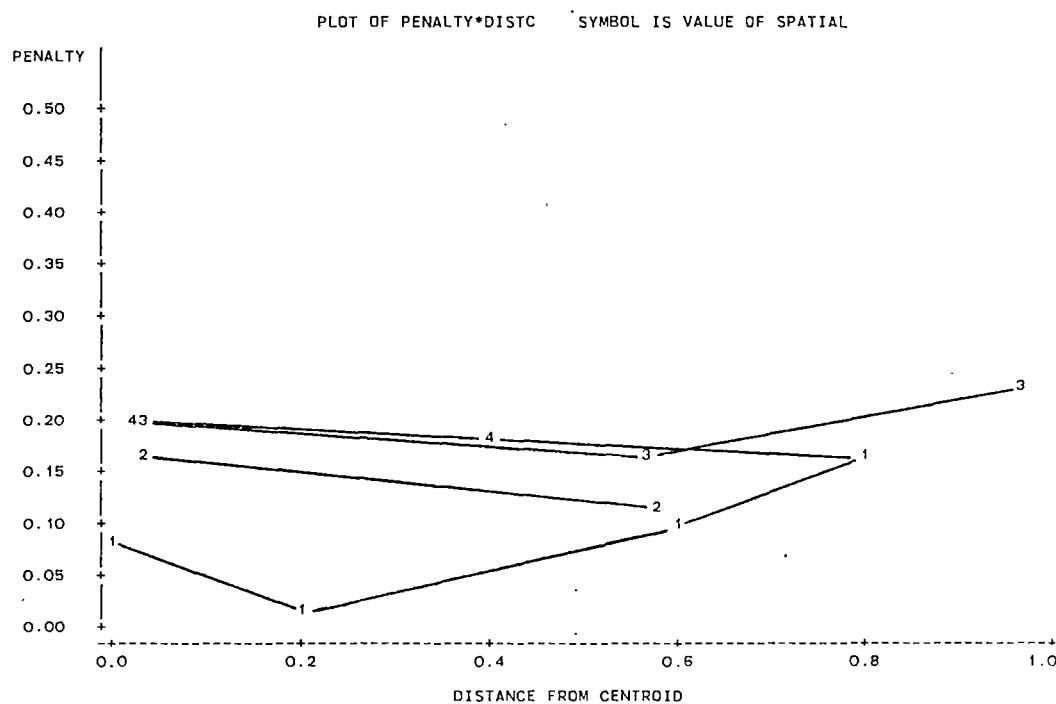


Figure 6.8. Penalties for Balancing Both Route-Load and Route-Length



(C) THIRD DEMAND PATTERN



(D) FOURTH DEMAND PATTERN

Figure 6.8. (Continued)

situations in which demand is actually uniform, but also for those situations in which route load is measured by the number of customers on a route (e.g., certain cases of the driver-sell environment).

3. Balancing of route lengths costs less if the spatial pattern of customer locations is relatively structured.
4. Distribution systems which currently balance route lengths can also balance route loads at little or no added routing cost if minor tradeoffs are made between the two objectives.

Further research, employing a large number of different problems, is required to test more completely the validity of these statements.

CHAPTER VII

SUMMARY, CONCLUSIONS, AND RECOMMENDATIONS

Summary

The purpose of this research was to examine the use of multi-criteria analysis in solving the workload balanced vehicle routing problem (WBVRP), a problem which was not found in the literature during an extensive search by the author. Four specific goals were established in Chapter I to accomplish this.

Goal Number One

The first goal was to develop a multiobjective model structure to solve the WBVRP, utilizing user interaction to make necessary tradeoffs among the three objectives of the problem. This model structure was developed in Chapter III. It is based on a heuristic implementation of the Method of Satisfactory Goals (Benson, 1975). Using this method, a problem is solved by minimizing a least satisfactory objective function at each stage, subject to satisfactory levels of the other objective functions being maintained.

Goal Number Two

The second goal was to develop and evaluate methods to minimize the three objective functions of the WBVRP. These algorithms were presented in Chapter IV. All three of these algorithms are based on arc exchange

heuristics. For the distance minimization algorithm, the arc exchanges are used to provide reductions in the total distance objective; for the workload deviation minimization algorithms, the arc exchanges are used to cluster customers into potential routes. Representative solution times for these algorithms on an IBM 3081D are shown in Table 7.1. For the distance minimization algorithm, solution times are highly dependent upon the number of customers in the problem. For the route-length deviation algorithm, the solution times are highly dependent upon the total number of TSPs solved by the algorithm. The solution times for the route-load deviation algorithm are not as dependent upon the number of TSPs solved. In either case, only about one-half of the variability in solution times for the deviation minimization algorithms could be explained by regression models.

The solution quality of the three algorithms as a function of the number of runs is also shown in Table 7.1. For the distance minimization algorithm, the measure of solution quality is expressed as a percentage difference from the best known solution. For the deviation minimization algorithms, the solution quality is expressed as a percentage of the maximum possible improvement in deviation.

Goal Number Three

The third goal was to incorporate the multiobjective model structure into an interactive computer program, and to evaluate the performance of the program in terms of efficiency (solution times) and effectiveness (solution values). The interactive computer program was presented in Chapter V. This program uses goal tradeoffs to estimate the effects of constraint relaxation on the minimization of the least

TABLE 7.1
SOLUTION TIMES AND SOLUTION QUALITY OF SINGLE-OBJECTIVE ALGORITHMS

Algorithm	Minimum Problem Size	CPU Sec.	Maximum Problem Size	CPU Sec.	Solution Quality		
					1 Run	2 Runs	3 Runs
Total Distance	22 cities	0.15	100 cities	17.91	3.75	3.17	3.14
Route-Length Deviation	315 TSPs	0.94	13,101 TSPs	16.10	91.22	93.40	94.72
Route-Load Deviation	2 TSPs	0.28	3,648 TSPs	5.39	82.83	83.93	92.92

satisfactory objective function. A convergence analysis using utility functions showed the program to yield final solutions which were close to one another when the procedure was begun with different starting solutions, although no identical solutions were generated. A nondominance analysis showed about fifteen percent of the solutions generated by the procedure to be dominated. The majority of problems were solved in less than one minute CPU time.

Goal Number Four

The fourth goal was to determine the penalty which must be paid by distribution managements in order to balance route lengths and route loads under differing patterns of demand and customer location. In Chapter VI, a standard 36-city problem was set up and solved using different depot locations, customer demand patterns, and customer location patterns. Penalties were calculated as a fraction of total routing costs without workload balancing.

Conclusions

The heuristic version of the Method of Satisfactory Goals as presented herein appears to offer good satisficing solutions to the WBVRP. The procedure is straightforward and easy to use, and the amount of user input at each step is minimal. Solution times are not excessive for the quality of solutions obtained. To improve the solution quality, the following are required:

1. Decrease the error of the single-objective algorithms by additional runs of the appropriate algorithm in each iteration.

2. Provide better estimates of the effect of constraint relaxation in the two constraining achievement levels.

However, these improvements could lead to substantially increased computing times.

From the analysis of workload balancing costs, only conclusions regarding the standard 36-city problem of Chapter VI can be made. More analyses must be performed before conclusions can be extended to WBVRPs in general. However, the analysis of the 36-city problem points to the likelihood that distribution systems with centrally located depots are candidates for workload balancing, particularly if the demand is constant. Moreover, it is likely that those distribution systems currently balancing route lengths can also balance route loads with little or no additional routing costs.

Recommendations for Further Research

Attempts to examine the effects of problem characteristics on the cost of workload balancing were only partially successful. Much more research is needed to identify the characteristics of a problem which make it a likely or unlikely candidate for workload balancing. In addition, specific cases need to be solved in a real world setting. These cases need to evaluate not only the effectiveness of the solution technique, but also the actual benefits derived from workload balancing.

Specific modifications to the procedure developed in this research can be suggested for evaluation. In particular, the following should be explored:

1. Use of an exact TSP procedure inside the deviation minimization routines instead of the 3-opt method employed herein. Because

of the time required by exact methods, an efficient way of filtering out unpromising exchanges could be employed to reduce the total number of TSPs to be solved.

2. Use of 4-arc exchanges instead of 3-arc exchanges in the deviation minimization clustering procedures to provide more types of customer exchanges between routes. To offset the greater number of arc combinations involved, the procedure could be modified to consider exchanges between only two routes at a time.
3. A different means of estimating the effect of relaxing one or both of the two constraining achievement levels at each iteration of the procedure.

Other multicriteria approaches to solving the WBVRP can be investigated. One such approach would be to formulate the problem as a multiple objective integer program, and then use heuristics to solve a single objective integer program in each iteration of a procedure such as described by Gabbiani and Magazine (1985).

A natural extension of the WBVRP is the workload balanced vehicle scheduling problem (WBVSP), in which temporal constraints are considered. The problem might involve time windows during which deliveries must be made, or might allow a longer period of time (e.g., a week) in which to balance the workload elements.

Deterministic demands have been assumed in all of the WBVRPs solved herein. Future research could involve stochastic versions of either the WBVRP or the WBVSP, to account for the variability of demand at each customer location.

Finally, research in the use of exact methods to solve this and

other vehicle routing problems could be undertaken. Early researchers found the computing times to obtain optimal solutions to be prohibitive, and resorted to the development of innovative heuristics to arrive at "good" solutions, instead. This approach has continued over the years, although computer technology has achieved CPU speeds many thousands of times faster than was available to those earlier researchers. Over the same period, computational costs have declined by a factor of several thousands. Efforts should be made to exploit these gains in computer technology by solving at least some aspects of vehicle routing problems in an optimal manner.

BIBLIOGRAPHY

- Balas, E. and N. Christofides. "A Restricted Lagrangean Approach to the Traveling Salesman Problem." Mathematical Programming, 21 (1981), 19-46.
- Balinski, M. and R. Quandt. "On an Integer Program for a Delivery Problem." Operations Research, 12 (1964), 300-304.
- Ball, M. O., B. L. Golden, A. A. Assad, L. D. Bodin. "Planning for Truck Fleet Size in the Presence of a Common-Carrier Option." Decision Sciences, 14 (1983), 103-120.
- Beasley, J. E. "Route First - Cluster Second Methods for Vehicle Routing." OMEGA, 11 (1983), 403-408.
- Bellman, R. "Dynamic Programming Treatment of the Travelling Salesman Problem." J. ACM, 9 (1962), 61-63.
- Bellmore, M. and J. Malone. "Pathology of Traveling-Salesman Subtour-Elimination Algorithms." Operations Research, 13 (1971), 278-307.
- Bellmore, M. and G. Nemhauser. "The Traveling Salesman Problem: A Survey." Operations Research, 16 (1974), 538-558.
- Beltrami, E. and L. Bodin. "Networks and Vehicle Routing for Municipal Waste Collection." Networks, 4 (1974), 65-94.
- Benayoun, R., J. de Montgolfier, J. Tergny, and O. Larichev. "Linear Programming with Multiple Objective Functions: Step Method (STEM)." Mathematical Programming, 1 (1971), 366-375.
- Benson, R. G. "Interactive Multiple Criteria Optimization Using Satisfactory Goals." (Ph.D. dissertation, University of Iowa, 1975.)
- Benton, W. C. "Evaluating a Modified Heuristic for the Multiple-Vehicle Scheduling Problem." IIE Transactions, 18 (1986), 70-78.
- Bodin, L. and L. Berman. "Routing and Scheduling of School Buses by Computer." Transportation Science, 13 (1979), 113-125.
- Bodin, L., B. Golden, A. Assad, M. Ball. "Routing and Scheduling of Vehicles and Crews." Computers and Operations Research, 10 (1983), 63-211.
- Bodin, L. and B. Golden. "Classification in Vehicle Routing and Scheduling." Networks, 11 (1981), 97-108.

- Boychuk, L. M. and V. O. Ovchinnikov. "Principal Methods for Solution of Multicriterial Optimization Problems (Survey)." Soviet Automatic Control, 6 (1973), 1-4.
- Chapleau, L., J. Ferland, J. M. Rousseau. "Clustering for Routing in Dense Areas." University of Montreal Transportation Research Center Publication No. 234, (1981).
- Charnes, A. and W. W. Cooper. Management Models and Industrial Applications of Linear Programming. New York: John Wiley, 1961.
- Cheshire, I. M., A. M. Malleson, P. F. Naccache. "A Dual Heuristic for Vehicle Scheduling." J. Operational Research Society, 33 (1982), 51-61.
- Christofides, N. "Bounds for the Travelling-Salesman Problem." Operations Research, 20 (1972), 1044-1056.
- Christofides, N. "The Travelling Salesman Problem." Combinatorial Optimization. Ed. N. Christofides, R. Mingozzi, P. Toth, and C. Sandi. New York: Wiley, 1979, pp. 131-149.
- Christofides, N. and S. Eilon. "An Algorithm for the Vehicle-Dispatching Problem." Operational Research Quarterly, 20 (1969), 309-318.
- Christofides, N. and S. Eilon. "Algorithms for Large-Scale Travelling Salesman Problems." Operational Research Quarterly, 23 (1972), 511-518.
- Christofides, N., A. Mingozzi, P. Toth. "Exact Algorithms for the Vehicle Routing Problem, Based on Spanning Tree and Shortest Path Relaxations." Mathematical Programming, 20 (1981), 255-282.
- Clarke, G. and J. W. Wright. "Scheduling of Vehicles from a Central Depot to a Number of Delivery Points." Operations Research, 12 (1964), 568-581.
- Croes, G. A. "A Method for Solving Traveling-Salesman Problems." Operations Research, 6 (1958), 791-812.
- Dantzig, G., D. Fulkerson, S. Johnson. "Solution of a Large-Scale Traveling Salesman Problem." Operations Research, 2 (1954) 393-410.
- Dantzig, G. and J. Ramser. "The Truck Dispatching Problem." Management Science, 6 (1959), 81-91.
- Dileepan, P. "Delivery Planning Problem." (Ph.D. dissertation, University of Houston, 1984.)
- Dyer, J. S. "Interactive Goal Programming." Management Science, 19 (1972), 62-70.
- Eastman, W. L. "Linear Programming with Pattern Constraints." (Ph.D. dissertation, Harvard University, 1958).

Eilon, S., C. Watson-Gandy, N. Christofides. Distribution Management: Mathematical Modeling and Practical Analysis. New York: Hafner, 1971.

Evans, S. R. and J. P. Norback. "An Heuristic Method for Solving Time-Sensitive Routing Problems." J. Operational Research Society, 35 (1984), 407-414.

Farquhar, P. H. "A Survey of Multiattribute Utility Theory and Applications." Multiple Criteria Decision Making. Ed. M. K. Starr and M. Zeleny. New York: North Holland, 1977, pp. 59-90.

Fichefet, J. "GPSTEM: An Interactive Multiobjective Optimization Method." Progress in Operation Research, Vol. 1. Ed. A. Prekopa. Amsterdam: North Holland, 1976, pp. 317-332.

Fishburn, P. C. "Lexicographic Orders, Utilities, and Decision Rules: A Survey." Management Science, 20 (1974), 1442-1471.

Fisher, M. "The Lagrangian Relaxation Method for Solving Integer Programming Problems." Management Science, 27 (1981), 1-12.

Fisher, M. and R. Jaikumar. "A Generalized Assignment Heuristic for Vehicle Routing." Networks, 11 (1981), 109-124.

Flood, M. "The Traveling Salesman Problem." Operations Research, 4 (1956), 61-75.

Foster, B. and D. Ryan. "An Integer Programming Approach to the Vehicle Scheduling Problem." Operational Research Quarterly, 27 (1976), 367-384.

Gabbani, D. and M. Magazine. "An Interactive Heuristic Approach for Multi-Objective Integer Programming Problems." Department of Management Sciences Working Paper, University of Waterloo, 1985.

Gal, T. and J. Nedoma. "Multiparametric Linear Programming." Management Science, 18 (1972), 406-421.

Gaskell, T. "Bases for Vehicle Fleet Scheduling." Operational Research Quarterly, 18 (1967), 281-295.

Geoffrion, A. M., J. S. Dyer, and A. Feinberg. "An Interactive Approach for Multi-Criterion Optimization, with an Application to the Operation of an Academic Department." Management Science, 19 (1972), 357-368.

Gillett, B. and J. Johnson. "Multi-terminal Vehicle-dispatch Algorithm." OMEGA, 4 (1976), 711-718.

Gillett, B. and L. Miller. "A Heuristic Algorithm for the Vehicle Dispatch Problem." Operations Research, 22 (1974), 340-349.

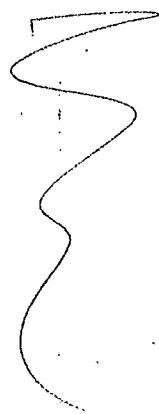
- Golden, B., A. Assad, L. Levy, F. Gheysens. "The Fleet Size and Mix Vehicle Routing Problem." Computers and Operations Research, 11 (1984), 49-66.
- Golden, B., L. Bodin, T. Doyle, W. Stewart. "Approximate Traveling Salesman Algorithms." Operations Research, 28 (1980), 694-711.
- Golden, B., T. Magnanti, H. Nguyen. "Implementing Vehicle Routing Algorithms." Networks, 7 (1977), 113-148.
- Haines, Y. Y., W. A. Hall, and H. T. Freedman. Multiobjective Optimization in Water Resources Systems, the Surrogate Worth Tradeoff Method. New York: Elsevier Scientific, 1975.
- Hansen, K. and J. Krarup. "Improvements of the Held-Karp Algorithm for the Symmetric Traveling Salesman Problem." Mathematical Programming, 7 (1974), 87-96.
- Held, M. and R. Karp. "A Dynamic Programming Approach to Sequencing Problems." J. SIAM, 10 (1962), 196-210.
- Held, M. and R. Karp. "The Traveling Salesman Problem and Minimum Spanning Trees." Operations Research, 18 (1970), 1138-1162.
- Held, M. and R. Karp. "The Traveling Salesman Problem and Minimum Spanning Trees - Part II." Mathematical Programming, 1 (1971), 6-25.
- Holmes, R. and R. Parker. "A Vehicle Scheduling Procedure Based Upon Savings and a Solution Perturbation Scheme." Operational Research Quarterly, 27 (1976), 83-92.
- Husban, Ahmad O. "Balancing Routes in a Class of Vehicle Routing Problems." (Ph.D. dissertation, Rensselaer Polytechnic Institute, 1985.)
- Hwang, Ching-Lai and Abu S. Masud. Multiple Objective Decision Making - Methods and Applications. Berlin: Springer-Verlag, 1979.
- Ignizio, J. P. Goal Programming and Extensions. Massachusetts: Lexington Books, 1976.
- Karg, L. and G. Thompson. "A Heuristic Approach to Solving Traveling Salesman Problems." Management Science, 10 (1964), 225-248.
- Keeny, R. L. "Utility Functions for Multiattributed Consequences." Management Science, 18 (1972), 276-287.
- Kirby, R. and J. McDonald. "The Savings Method for Vehicle Scheduling." Operational Research Quarterly, 24 (1973), 305.
- Klein, D. and E. Hannan. "An Algorithm for the Multiple Objective Integer Linear Programming Problem." European J. of Operational Research, 9 (1982), 378-385.

- Krolak, P., W. Felts, J. Nelson. "A Man-Machine Approach Toward Solving the Generalized Truck Dispatching Problem." Transportation Science, 6 (1972), 149-170.
- Lawrence, J. L. "Interactive Vehicle Dispatching: A Hybrid Approach." (Ph.D. dissertation, University of Missouri - Rolla, 1981.)
- Lee, S. "Interactive Integer Goal Programming: Methods and Application." Multiple Criteria Problem Solving. Ed. S. Zionts. New York: Springer-Verlag, 1978, pp. 362-383.
- Lin, S. "Computer Solutions of the Traveling Salesman Problem." Bell System Technical Journal, 44 (1965), 2245-2269.
- Lin, S. and B. Kernighan. "An Effective Heuristic Algorithm for the Traveling Salesman Problem." Operations Research, 21 (1973), 498-516.
- Little, J., K. Murty, D. Sweeny, C. Karel. "An Algorithm for the Traveling Salesman Problem." Operations Research, 11 (1963), 972-989.
- Martin, G. T. "An Accelerated Euclidean Algorithm for Integer Linear Programming." Recent Advances in Mathematical Programming. Eds. R. L. Graves and P. Wolfe. 1963, pp. 311-318.
- Miliotis, P. "Integer Programming Approaches to the Traveling Salesman Problem." Mathematical Programming, 10 (1976), 367-378.
- Miller, C., A. Tucker, R. Zemlin. "Integer Programming Formulation of Travelling Salesman Problems." J. ACM, 7 (1960), 326-332.
- Mole, R. "A Survey of Local Delivery: Vehicle Routing Methodology." J. Operational Research Society, 30 (1979), 245-252.
- Mole, R. and S. Jameson. "A Sequential Route-Building Algorithm Employing a Generalized Savings Criterion." Operational Research Quarterly, 27 (1976), 503-511.
- Newton, R. and W. Thomas. "Bus Routing in a Multi-School System." Computers and Operations Research, 1 (1974), 213-222.
- Norback, J. and R. Love. "Geometric Approaches to Solving the Traveling Salesman Problem." Management Science, 23 (1977), 1208-1223.
- Park, Y. "Solutions to Vehicle Routing Problems in a Multiple Criteria Environment." (Ph.D. dissertation, Oklahoma State University, Stillwater, 1984).
- Pierce, J. "Direct Search Algorithms for Truck-Dispatching Problems Part I." Transportation Research, 3 (1969), 1-42.

- Reeves, G. R. and L. S. Franz. "A Simplified Interactive Multiple Objective Linear Programming Procedure." Computers and Operations Research, 12 (1985), 589-601.
- Reiter, S. and G. Sherman. "Discrete Optimizing." J. SIAM, 13 (1965), 864-889.
- Rosenkrantz, D., R. Stearns, P. Lewis. "Approximate Algorithms for the Traveling Salesperson Problem." Proc. of 15th Annual IEEE Symposium on Switching and Automata Theory, (1974), pp. 33-42.
- Russell, R. "An Effective Heuristic for the M-tour Traveling Salesman Problem with Some Side Conditions." Operations Research, 25 (1977), 517-524.
- Shapiro, D. "Algorithms for the Solution of the Optimal Cost Travelling Salesman Problem." (Ph.D. dissertation, Washington University, St. Louis, 1966.)
- Shapiro, Jeremy F. Mathematical Programming: Structures and Algorithms. New York: Wiley, 1979.
- Stacey, P. "Practical Vehicle Routing Using Computer Programs." J. Operational Research Society, 34 (1983), 975-981.
- Steuer, R. "An Interactive Multiple Objective Linear Programming Procedure." TIMS Studies in the Management Sciences, 6 (1977), 225-239.
- Stewart, W. "New Algorithms for Deterministic and Stochastic Vehicle Routing Problems." (D.B.A. dissertation, University of Maryland, 1981.)
- Stewart, W. "An Accelerated Branch Exchange Heuristic for the Traveling Salesman Problem." School of Business Administration Working Paper No. 85-004, College of William and Mary, Williamsburg, 1985.
- Stewart, W. and B. Golden. "A Lagrangean Relaxation Heuristic for Vehicle Routing." European J. of Operational Research, 15 (1984), 84-88.
- Tillman, F. and T. Cain. "An Upper Bounding Algorithm for the Single and Multiple Terminal Delivery Problem." Management Science, 18 (1972), 664-682.
- Tillman, F. and H. Cochran. "A Heuristic Approach for Solving the Delivery Problem." J. of Industrial Engineering, 19 (1968), 354-358.
- Turner, W., P. Ghare, L. Foulds. "Transportation Routing Problem - A Survey." AIEE Transactions, 6 (1974), 288-301.
- Tyagi, M. "A Practical Method for the Truck Dispatching Problem." J. Operations Research Society of Japan, 10 (1968), 76-92.

- Waters, C. G. J. "Interactive Vehicle Routing." J. Operational Research Society, 35 (1984), 821-826.
- Webb, M. "Some Methods of Producing Approximate Solutions to Traveling Salesman Problems with Hundreds or Thousands of Cities." Operational Research Quarterly, 22 (1971), 49-66.
- Williams, B. W. "Vehicle Scheduling: Proximity Priority Scheduling." J. Operational Research Society, 33 (1982), 961-966.
- Wiorkowski, J. and K. McElvain. "A Rapid Heuristic Algorithm for the Approximate Solution of the Traveling Salesman Problem." Transportation Research, 9 (1975), 181-185.
- Wren, A. and A. Holliday. "Computer Scheduling of Vehicles from One or More Depots to a Number of Delivery Points." Operational Research Quarterly, 23 (1972), 333-344.
- Yellow, P. "A Computational Modification to the Savings Method of Vehicle Scheduling." Operational Research Quarterly, 21 (1970), 281-283.
- Yu, P. L. and M. Zeleny. "The Set of all Non-Dominated Solutions in Linear Cases and a Multicriteria Simplex Method." Journal of Mathematical Analysis and Applications, 49 (1975), 430-468.
- Zeleny, M. Multiple Criteria Decision Making. New York: McGraw-Hill, 1982.
- Zionts, S. "Integer Linear Programming with Multiple Objectives." Annals of Discrete Mathematics, 1 (1977), 551-562.
- Zionts, S. and J. Wallenius. "An Interactive Programming Method for Solving the Multiple Criteria Problem." Management Science, 22 (1976), 652-663.

APPENDIXES



APPENDIX A

**APPROXIMATE NUMBER OF QUALIFIED ARC COMBINATIONS
IN DEVIATION MINIMIZATION ALGORITHMS**

APPROXIMATE NUMBER OF QUALIFIED ARC COMBINATIONS
IN DEVIATION MINIMIZATION ALGORITHMS

There are $\binom{N+R}{3}$ total arc combinations in a 3-arc exchange algorithm, where N is the number of customers in the problem and R is the number of routes. To minimize the maximum deviation between routes, only certain of these combinations are 'qualified' for consideration. To be qualified, a combination (1) must have at least one arc in the largest route or in the smallest route, and (2) cannot have all three arcs in the same route. To find an approximate number of qualified arc combinations, all routes are assumed to be of about the same size so that the probability of a single arc being in a given route is $\frac{1}{R}$. The easiest way of finding the number of qualified combinations is to first find the number of unqualified combinations and subtract that number from the total number of combinations. A combination is unqualified for consideration if any of the following apply: (1) all three arcs are in the smallest route, (2) all three arcs are in the largest route, or (3) all three arcs are contained elsewhere.

First, the total number of combinations is found.

$$\text{Combinations (total)} = \binom{N+R}{3} \quad (\text{A.1})$$

$$\text{or} \quad \text{Combinations (total)} = \frac{(N+R)(N+R-1)(N+R-2)}{6} \quad (\text{A.2})$$

which, if $N+R$ is sufficiently large, can be approximated by

$$\text{Combinations (total)} = \frac{(N+R)^3}{6} \quad (\text{A.3})$$

Next, the total number of combinations of three arcs in the large route is found.

$$\text{Combinations (3 in large route)} = \frac{\left(\frac{N}{R}+1\right)}{3} \quad (\text{A.4})$$

$$\text{or Combinations (3 in large route)} = \left(\frac{N}{R}+1\right)\left(\frac{N}{R}\right)\left(\frac{N}{R}-1\right) \quad (\text{A.5})$$

$$\text{or Combinations (3 in large route)} = \frac{N^3 - NR^2}{6R^3}. \quad (\text{A.6})$$

Since it is assumed that routes are of approximately equal size, the number of combinations of three arcs in the small route is also given by (A.6). Therefore

$$\text{Combinations (3 in large or small route)} = \frac{2(N^3 - NR^2)}{6R^3}. \quad (\text{A.7})$$

Now the total number of combinations of three arcs contained outside of the large or small routes is given by

$$\text{Combinations (3 elsewhere)} = \frac{(N+R-2)\left(\frac{N}{R}+1\right)}{3} \quad (\text{A.8})$$

$$\text{or Combinations (3 elsewhere)} = \frac{(N+R-\frac{2N}{R}-2)}{3} \quad (\text{A.9})$$

$$\text{or Combinations (3 elsewhere)} = \frac{(N+R-\frac{2N}{R}-2)(N+R-\frac{2N}{R}-3)(N+R-\frac{2N}{R}-4)}{6} \quad (\text{A.10})$$

Once again, assuming $N+R$ sufficiently large, the equation above can be approximated by

$$\text{Combinations (3 elsewhere)} = \frac{(NR+R^2-2N)^3}{6R^3}. \quad (\text{A.11})$$

Now, the number of qualified combinations is given by

$$Q = \text{Combinations (total)} - \text{Combinations (3 in large or small route)} \\ - \text{Combinations (3 elsewhere)}$$

$$\text{or } Q = \frac{(N+R)^3}{6} - \frac{2(N^3-NR^2)}{6R^3} - \frac{(NR+R^2-2N)^3}{6R^3}. \quad (\text{A.12})$$

APPENDIX B

VEHICLE ROUTING PROBLEM DATA

TABLE B.1
GASKELL'S 22-CITY PROBLEM

City	X	Y	Demand	City	X	Y	Demand
1	295	272	125	12	267	242	300
2	301	258	84	13	259	265	250
3	309	260	60	14	315	233	500
4	217	274	500	15	329	252	150
5	218	278	300	16	318	252	100
6	282	267	175	17	329	224	250
7	242	249	350	18	267	213	120
8	230	262	150	19	275	192	600
9	249	268	1100	20	303	201	500
10	256	267	4100	21	208	217	175
11	265	257	225	22	326	181	75

Vehicle Capacity = 4500

Maximum Miles = 240

Allowance = 10 miles

Depot Coordinates: 266, 235

TABLE B.2
GASKELL'S 29-CITY PROBLEM

City	X	Y	Demand	City	X	Y	Demand
1	218	382	300	16	119	357	150
2	218	358	3100	17	115	341	100
3	201	370	125	18	153	351	150
4	214	371	100	19	175	363	400
5	224	370	200	20	180	360	300
6	210	382	150	21	159	331	1500
7	104	354	150	22	188	357	100
8	126	338	450	23	152	349	300
9	119	340	300	24	215	389	500
10	129	349	100	25	212	394	800
11	126	347	950	26	188	393	300
12	125	346	125	27	207	406	100
13	116	355	150	28	184	410	150
14	126	335	150	29	207	392	1000
15	125	355	550				

Vehicle Capacity = 4500

Maximum Miles = 240

Allowance = 10 Miles

Depot Coordinates: 162, 354

TABLE B.3
GASKELL'S 32-CITY PROBLEM

City	X	Y	Demand	City	X	Y	Demand
1	298	427	700	17	297	410	550
2	309	445	400	18	315	407	650
3	307	464	400	19	314	406	200
4	336	475	1200	20	321	391	400
5	320	439	40	21	321	398	300
6	321	437	80	22	314	394	1300
7	322	437	2000	23	313	378	700
8	323	433	900	24	304	382	750
9	324	433	600	25	295	402	1400
10	323	429	750	26	283	406	4000
11	314	435	1500	27	279	399	600
12	311	442	150	28	271	401	1000
13	304	427	250	29	264	414	500
14	293	421	1600	30	277	439	2500
15	296	418	450	31	290	434	1700
16	261	384	700	32	319	433	1100

Vehicle Capacity = 8000

Maximum Miles = 240

Allowance = 10 Miles

Depot Coordinates: 292, 425

TABLE B.4
CHRISTOFIDES AND EILON'S 50-CITY PROBLEM

City	X	Y	Demand	City	X	Y	Demand
1	37	52	7	26	27	68	7
2	49	49	30	27	30	48	15
3	52	64	16	28	43	67	14
4	20	26	9	29	58	48	6
5	40	30	21	30	58	27	19
6	21	47	15	31	37	69	11
7	17	63	19	32	38	46	12
8	31	62	23	33	46	10	23
9	52	33	11	34	61	33	26
10	51	21	5	35	62	63	17
11	42	41	19	36	63	69	6
12	31	32	29	37	32	22	9
13	5	25	23	38	45	35	15
14	12	42	21	39	59	15	14
15	36	16	10	40	5	6	7
16	52	41	15	41	10	17	27
17	27	23	3	42	21	10	13
18	17	33	41	43	5	64	11
19	13	13	9	44	30	15	16
20	57	58	28	45	39	10	10
21	62	42	8	46	32	39	5
22	42	57	8	47	25	32	25
23	16	57	16	48	25	55	17
24	8	52	10	49	48	28	18
25	7	38	28	50	56	37	10

Vehicle Capacity = 160

Maximum Miles: None

Depot Coordinates: 30, 40

TABLE B.5
CHRISTOFIDES AND EILON'S 75-CITY PROBLEM

No.	x	y	q												
1	22	22	18	20	66	14	22	39	30	60	16	58	40	60	21
2	36	26	26	21	44	13	28	40	30	50	33	59	70	64	24
3	21	45	11	22	26	13	12	41	12	17	15	60	64	4	13
4	45	35	30	23	11	28	6	42	15	14	11	61	36	6	15
5	55	20	21	24	7	43	27	43	16	19	18	62	30	20	18
6	33	34	19	25	17	64	14	44	21	48	17	63	20	30	11
7	50	50	15	26	41	46	18	45	50	30	21	64	15	5	28
8	55	45	16	27	55	34	17	46	51	42	27	65	50	70	9
9	26	59	29	28	35	16	29	47	50	15	19	66	57	72	37
10	40	66	26	29	52	26	13	48	48	21	20	67	45	42	30
11	55	65	37	30	43	26	22	49	12	38	5	68	38	33	10
12	35	51	16	31	31	76	25	50	15	56	22	69	50	4	8
13	62	35	12	32	22	53	28	51	29	39	12	70	66	8	11
14	62	57	31	33	26	29	27	52	54	38	19	71	59	5	3
15	62	24	8	34	50	40	19	53	55	57	22	72	35	60	1
16	21	36	19	35	55	50	10	54	67	41	16	73	27	24	6
17	33	44	20	36	54	10	12	55	10	70	7	74	40	20	10
18	9	56	13	37	60	15	14	56	6	25	26	75	40	37	20
19	62	48	15	38	47	66	24	57	65	27	14				

q = demand in cwt

Vehicle Capacity = 7 tons

Maximum miles: None

Depot Coordinates: 40, 40

TABLE B.6
CHRISTOFIDES AND EILON'S 100-CITY PROBLEM

No.	x	y	q												
1	41	49	10	26	45	30	17	51	49	58	10	76	49	42	13
2	35	17	7	27	35	40	16	52	27	43	9	77	53	43	14
3	55	45	13	28	41	37	16	53	37	31	14	78	61	52	3
4	55	20	19	29	64	42	9	54	57	29	18	79	57	48	23
5	15	30	26	30	40	60	21	55	63	23	2	80	56	37	6
6	25	30	3	31	31	52	27	56	53	12	6	81	55	54	26
7	20	50	5	32	35	69	23	57	32	12	7	82	15	47	16
8	10	43	9	33	53	52	11	58	36	26	18	83	14	37	11
9	55	60	16	34	65	55	14	59	21	24	28	84	11	31	7
10	30	60	16	35	63	65	8	60	17	34	3	85	16	22	41
11	20	65	12	36	2	60	5	61	12	24	13	86	4	18	35
12	50	35	19	37	20	20	8	62	24	58	19	87	28	18	26
13	30	25	23	38	5	5	16	63	27	69	10	88	26	52	9
14	15	10	20	39	60	12	31	64	15	77	9	89	26	35	15
15	30	5	8	40	40	25	9	65	62	77	20	90	31	67	3
16	10	20	19	41	42	7	5	66	49	73	25	91	15	19	1
17	5	30	2	42	24	12	5	67	67	5	25	92	22	22	2
18	20	40	12	43	23	3	7	68	56	39	36	93	18	24	22
19	15	60	17	44	11	14	18	69	37	47	6	94	26	27	27
20	45	65	9	45	6	38	16	70	37	56	5	95	25	24	20
21	45	20	11	46	2	48	1	71	57	68	15	96	22	27	11
22	45	10	18	47	8	56	27	72	47	16	25	97	25	21	12
23	55	5	29	48	13	52	36	73	44	17	9	98	19	21	10
24	65	35	3	49	6	68	30	74	46	13	8	99	20	26	9
25	65	20	6	50	47	47	13	75	49	11	18	100	18	18	17

q = demand in cwt.

Vehicle Capacity = 10 tons

Maximum Miles: None

Depot Coordinates: 35, 35

TABLE B.7
EXAMPLE 33-CITY PROBLEM

City	X	Y	Demand	City	X	Y	Demand
1	3	3	15	18	24	14	10
2	8	5	31	19	25	18	15
3	16	4	10	20	15	18	27
4	6	8	9	21	4	20	35
5	12	7	40	22	11	20	15
6	24	7	35	23	20	20	22
7	9	10	14	24	3	22	6
8	14	10	10	25	8	22	25
9	20	10	3	26	13	22	41
10	21	11	17	27	15	23	30
11	3	13	15	28	18	22	43
12	5	14	26	29	5	24	10
13	5	15	20	30	8	26	23
14	12	14	15	31	15	27	39
15	10	16	11	32	20	25	10
16	12	17	5	33	25	25	14
17	18	16	17				

Vehicle Capacity = 150

Maximum Miles = 50

Depot Coordinates: 13, 16

APPENDIX C

DATA FOR REGRESSION MODELS

TABLE C.1
DATA FOR 3-OPT SOLUTION TIMES
REGRESSION MODEL

No. of Cities	CPU Seconds
22	0.115
22	0.203
22	0.237
22	0.169
22	0.191
22	0.138
29	0.316
29	0.488
29	0.491
29	0.381
29	0.456
29	0.267
32	0.339
32	0.340
32	0.313
32	0.323
32	0.526
32	0.345
50	2.980
50	1.482
50	3.031
50	2.801
50	2.832
50	3.444
75	5.430
75	5.930
75	5.300
75	6.304
75	5.250
75	5.400
100	16.070
100	29.990
100	15.770
100	28.440
100	21.270
100	16.090

TABLE C.2
DATA FOR ROUTE-LENGTH DEVIATION REGRESSION MODELS

N	R	B _{Ln}	B _{Ld}	RLX _{dist}	RLX _{Ld}	TSPs	CPU
22	5	.42	.96	.03	.001	7949	4.05
22	5	.42	.96	.03	.27	8220	4.19
22	5	.42	.96	.03	.64	7530	4.18
22	5	.42	.96	.03	.99	7533	4.16
22	5	.21	.82	.001	.001	3540	1.78
22	5	.21	.82	.001	.60	7047	3.46
22	5	.21	.82	.15	.41	11373	5.45
22	5	.21	.82	.15	.001	13737	6.73
29	4	.54	.29	.04	.001	5507	7.56
29	4	.54	.29	.04	.30	7060	9.77
29	4	.54	.29	.04	.77	7750	10.79
29	4	.54	.29	.04	.99	7775	10.86
29	4	.06	.36	.001	.001	2399	2.90
29	4	.06	.36	.001	.69	7235	10.01
29	4	.06	.36	.15	.30	5402	6.71
29	4	.06	.36	.15	.001	2758	4.14
32	4	.22	.22	.04	.001	2608	3.88
32	4	.22	.22	.04	.22	3226	4.88
32	4	.22	.22	.04	.53	5149	6.12
32	4	.22	.22	.04	.99	7724	7.74
32	4	.09	.09	.001	.001	539	.98
32	4	.09	.09	.001	.53	1057	1.72
32	4	.09	.09	.10	.10	1542	2.45
32	4	.09	.09	.10	.001	868	1.66
50	5	.45	.05	.001	.001	883	2.42
50	5	.45	.05	.001	.28	2322	6.05
50	5	.45	.05	.001	.61	2322	6.06
50	5	.45	.05	.001	.99	2322	6.07
50	5	.09	.03	.001	.001	315	.94
50	5	.09	.03	.001	.30	1338	3.26
50	5	.09	.03	.20	.10	1338	3.26
50	5	.09	.03	.20	.001	315	.94
75	10	.72	.06	.05	.001	3750	5.06
75	10	.72	.06	.05	.31	3357	3.82
75	10	.72	.06	.05	.69	3360	3.85
75	10	.72	.06	.05	.99	3357	3.50
75	10	.63	.06	.001	.001	2151	2.59
75	10	.23	.05	.001	.001	3227	4.31
75	10	.23	.05	.001	.50	4737	5.92
75	10	.23	.05	.20	.10	13101	16.10
75	10	.23	.05	.20	.001	6628	8.81

TABLE C.3
DATA FOR ROUTE-LOAD DEVIATION REGRESSION MODELS

N	R	B _{Ln}	B _{Ld}	RLX _{dist}	RLX _{Ln}	TSPs	CPU
22	5	.42	.96	.001	.001	3156	1.81
22	5	.42	.96	.15	.001	1265	.99
22	5	.37	.62	.001	.58	257	.22
22	5	.37	.62	.08	.99	362	.47
22	5	.21	.82	.001	.001	2041	1.03
22	5	.03	.79	.05	.18	1293	.98
22	5	.02	.78	.03	.19	2011	1.44
29	4	.29	.54	.15	.001	1377	1.79
29	4	.29	.54	.001	.001	3648	5.39
29	4	.06	.37	.01	.32	1787	2.51
29	4	.06	.36	.01	.33	1667	2.35
29	4	.06	.36	.001	.001	2635	3.52
29	4	.03	.48	.13	.03	3107	4.77
29	4	.03	.35	.13	.02	2418	4.14
29	4	.06	.23	.001	.06	1231	1.55
29	4	.06	.23	.08	.12	1231	1.55
32	4	.22	.20	.001	.001	1140	1.47
32	4	.09	.11	.01	.21	93	.37
32	4	.05	.15	.001	.24	28	.16
32	4	.09	.11	.001	.001	539	.96
32	4	.09	.11	.001	.001	490	.96
32	4	.03	.10	.01	.05	314	.75
32	4	.04	.09	.05	.04	75	.29
32	4	.22	.20	.15	.001	479	.95
32	4	.06	.10	.001	.06	451	.82
32	4	.06	.10	.08	.13	451	.82
50	5	.09	.03	.001	.99	5	.31
50	5	.07	.05	.01	.99	66	.61
50	5	.09	.03	.001	.001	94	.48
50	5	.08	.06	.001	.01	182	.70
50	5	.08	.06	.20	.01	107	.51
50	5	.09	.03	.20	.001	94	.48
50	5	.45	.05	.001	.001	278	1.02
50	5	.45	.05	.15	.001	23	.37
50	5	.45	.01	.001	.80	7	.29
50	5	.45	.01	.08	.99	2	.28
75	10	.37	.06	.001	.92	230	1.29
75	10	.63	.06	.001	.98	178	1.12
75	10	.63	.06	.001	.001	465	1.10
75	10	.23	.05	.001	.001	306	1.12
75	10	.23	.05	.001	.001	306	1.53
75	10	.10	.03	.09	.14	2	1.05

TABLE C.3 (Continued)

N	R	B _{Ln}	B _{Ld}	RLX _{dist}	RLX _{Ln}	TSPs	CPU
75	10	.23	.05	.20	.001	30	1.29
75	10	.72	.06	.001	.001	26	.97
75	10	.72	.04	.15	.001	19	1.30
75	10	.67	.02	.001	.99	1	1.04
75	10	.67	.02	.08	.99	1	1.04

FORTRAN PROGRAM LISTING

APPENDIX D

```

C                                         00000010
C                                         00000020
C                                         00000030
C***** 00000040
C                                         00000050
C                                         00000060
C                                         00000070
C
C   FORTRAN PROGRAM TO SOLVE THE WORKLOAD      00000080
C       BALANCED VEHICLE ROUTING PROBLEM        00000090
C       (WVRP), USING A HEURISTIC VERSION      00000100
C       OF THE METHOD OF SATISFACTORY GOALS.    00000110
C                                         00000120
C                                         00000130
C***** 00000140
C                                         00000150
C                                         00000160
C                                         00000170
C
C   AUTHOR:      J. D. ALLISON                 00000180
C   ADVISOR:     M. P. TERRELL                00000190
C   COMPUTER:    IBM 3081D                   00000200
C   DATE:        JUNE, 1986                  00000210
C                                         00000220
C                                         00000230
C
C   SCHOOL OF INDUSTRIAL ENGINEERING AND MANAGEMENT 00000240
C           OKLAHOMA STATE UNIVERSITY          00000250
C           STILLWATER, OKLAHOMA 74078         00000260
C                                         00000270
C                                         00000280
C***** 00000290
C                                         00000300
C                                         00000310
C                                         00000320
C
C   SUBROUTINES USED IN PROGRAM:               00000330
C                                         00000340
C
C   SAVNGS - CLARKE AND WRIGHT SAVINGS ALGORITHM. 00000350
C   TWOOPT - 2-OPT DISTANCE MINIMIZATION ALGORITHM. 00000360
C   THROPT - 3-OPT DISTANCE MINIMIZATION ALGORITHM. 00000370
C   LDDV2 - 2-ARC ROUTE LOAD DEVIATION ALGORITHM. 00000380
C   LDDV3 - 3-ARC ROUTE LOAD DEVIATION ALGORITHM. 00000390
C   LNDV2 - 2-ARC ROUTE LENGTH DEVIATION ALGORITHM. 00000400
C   LNDV3 - 3-ARC ROUTE LENGTH DEVIATION ALGORITHM. 00000410
C   FEAS2 - DETERMINES FEASIBILITY OF 2-ARC EXCHANGE. 00000420
C   XCHNG2 - PERFORMS 2-ARC EXCHANGE.            00000430
C   XCHNG3 - PERFORMS 3-ARC EXCHANGE.            00000440
C   FXCH2 - PERFORMS TEMPORARY 2-ARC EXCHANGE.   00000450
C   FXCH3 - PERFORMS TEMPORARY 3-ARC EXCHANGE.   00000460
C   TSP - 3-OPT VERSION OF TRAVELING SALESMAN PROBLEM ALGORITHM. 00000470
C   NONDOM - ELIMINATES DOMINATED TRADEOFFS FROM TRADEOFF ARRAY. 00000480
C   ADJUST - ACCEPTS AND EVALUATES MANUAL ADJUSTMENTS TO ROUTE. 00000490
C   BKTRAK - BACKTRACKS PROCEDURE TO A PRIOR SOLUTION.        00000500
C   DISPLA - DISPLAYS A PRIOR SOLUTION.          00000510
C   LOCK - LOCKS OUT A ROUTE FROM CALCULATIONS OF ROUTE-LOAD 00000520
C               DEVIATION AND ROUTE-LENGTH DEVIATION.        00000530
C                                         00000540
C                                         00000550
C
C   IMSL ROUTINES:                            00000560
C                                         00000570
C
C   GGUBFS - UNIFORM RANDOM NUMBER GENERATOR. 00000580
C   GGUD - UNIFORM RANDOM VECTOR GENERATOR.. 00000590
C   VSORA - MODIFIED QUICKSORT ALGORITHM.    00000600
C                                         00000610
C                                         00000620
C
C   HIERARCHY OF SUBROUTINE CALLS:            00000630
C                                         00000640
C                                         00000650
C
C   MAIN    CALLS SAVNGS, TWOOPT, THROPT, LDDV2, LDDV3, LNDV2, LNDV3, 00000660
C           NONDOM, TSP, ADJUST, BKTRAK, DISPLA, LOCK , GGUD, 00000670
C           VSORA                                         00000680
C                                         00000690
C
C   TWOOPT CALLS FEAS2, XCHNG2, GGUBFS        00000700
C   THROPT CALLS FEAS2, XCHNG2, XCHNG3, GGUBFS 00000710

```

```

C      LDDV2  CALLS FXCH2, TSP, XCHNG2, GGUBFS          00000720
C      LDDV3  CALLS FXCH2, FXCH3, TSP, XCHNG2, XCHNG3, GGUBFS 00000730
C      LNDV2  CALLS FXCH2, TSP, XCHNG2, GGUBFS          00000740
C      LNDV3  CALLS FXCH2, FXCH3, TSP, XCHNG2, XCHNG3, GGUBFS 00000750
C
C
C
C      GRAPHICS CAPABILITY PROVIDED BY TEKTRONIX PLOT 10 TERMINAL 00000760
C      CONTROL SYSTEM.                                         00000770
C
C
C***** VARIABLE DEFINITIONS:                                00000780
C
C
C      ALLOW      - DROP ALLOWANCE, MEASURED IN DISTANCE UNITS, AT EACH 00000820
C      NODE VISITED.                                         00000830
C
C      ANSWER     - CHARACTER VARIABLE, 'Y' OR 'N', IN RESPONSE TO 00000840
C      TERMINAL QUERY.                                         00000850
C
C      BACK       - O,1 VARIABLE INDICATING BACKWARD TRAVERSAL THROUGH 00000860
C      NETWORK IF VALUE IS 1.                               00000870
C
C      BACTRT     - NUMBER OF ACTIVE ROUTES (I.E., ROUTES WITH AT LEAST 00000880
C      1 CUSTOMER) IN THE BEST SOLUTION FOUND THUS FAR IN
C      MULTIPLE SOLUTIONS. SEE IACTRT.                     00000890
C
C      BCUMLD(I)  - CUMULATIVE LOAD AT NODE I IN BEST SOLUTION FOUND THUS 00000900
C      FAR. SEE CUMLD(I).                                 00000910
C
C      BCUMLN(I)  - CUMULATIVE DISTANCE AT NODE I IN BEST SOLUTION FOUND 00000920
C      THUS FAR. SEE CUMLN(I).                           00000930
C
C      BDEPOT(I)  - DEPOT OF ROUTE I IN BEST SOLUTION FOUND THUS FAR. 00000940
C      SEE DEPOT(I).                                     00000950
C
C      BESTDS     - MINIMUM TOTAL DISTANCE FOUND THUS FAR IN MULTIPLE 00000960
C      SOLUTIONS OF DISTANCE MINIMIZATION PROBLEM.        00000970
C
C      BESTLD(I)  - TOTAL LOAD OF ROUTE I IN BEST SOLUTION FOUND THUS 00000980
C      FAR. SEE LOAD(I).                                00000990
C
C      BESTLN(I)  - TOTAL LENGTH OF ROUTE I IN BEST SOLUTION FOUND THUS 00001000
C      FAR. SEE LENGTH(I).                            00001010
C
C      BESTRT     - NUMBER OF ROUTES IN BEST SOLUTION FOUND THUS FAR. 00001020
C      SEE IROUTE.                                     00001030
C
C      BESTP(I)   - PREDECESSOR OF NODE I IN BEST SOLUTION FOUND THUS FAR. 00001040
C      SEE PRED(I).                                    00001050
C
C      BESTS(I)   - SUCCESSOR OF NODE I IN BEST SOLUTION FOUND THUS FAR. 00001060
C      SEE SUCC(I).                                    00001070
C
C      BESTTR(I)  - ROUTE (TRUCK) ASSIGNED TO NODE I IN BEST SOLUTION 00001080
C      FOUND THUS FAR. SEE TRUCK(I).                  00001090
C
C      BLDDEV     - ROUTE LOAD DEVIATION IN BEST SOLUTION FOUND THUS FAR. 00001100
C      SEE LDDEV.                                     00001110
C
C      BLNDEV     - ROUTE LENGTH DEVIATION IN BEST SOLUTION FOUND THUS FAR. 00001120
C      SEE LNDEV.                                     00001130
C
C      BMAXLD     - MAXIMUM ROUTE LOAD IN BEST SOLUTION FOUND THUS FAR. 00001140
C      SEE MAXLD.                                    00001150
C
C      BMINLD     - MINIMUM ROUTE LOAD IN BEST SOLUTION FOUND THUS FAR. 00001160
C      SEE MINLD.                                    00001170
C
C      BMAXLN     - MAXIMUM ROUTE LENGTH IN BEST SOLUTION FOUND THUS FAR. 00001180
C      SEE MAXLN.                                    00001190
C
C      BMINLN     - MINIMUM ROUTE LENGTH IN BEST SOLUTION FOUND THUS FAR. 00001200
C      SEE MINLN.                                    00001210
C
C      BNTRAD     - NUMBER OF ORIGINAL TRADEOFFS FOUND IN FINAL ITERATION 00001220
C      OF BEST SOLUTION FOUND THUS FAR. SEE NTRADE.      00001230
C
C      BTRADE(,)  - ARRAY OF TRADEOFFS FOUND IN FINAL ITERATION OF BEST 00001240
C      SOLUTION FOUND THUS FAR. SEE TRADE(,).           00001250
C
C      CITY       - NODE INDEX USED IN READING PROBLEM DATA FROM FILE. 00001260
C
C      CLRNDX()   - VECTOR OF COLOR INDEXES USED BY GRAPHICS TERMINAL 00001270
C      PROGRAM.                                       00001280
C
C      CNSTR1     - CHARACTER VARIABLE CONTAINING DESCRIPTION OF FIRST 00001290
C      CONSTRAINING ACHIEVEMENT LEVEL.                 00001300
C
C      CNSTR2     - CHARACTER VARIABLE CONTAINING DESCRIPTION OF SECOND 00001310
C      CONSTRAINING ACHIEVEMENT LEVEL.                 00001320
C
C      CUMLD(I)   - CUMULATIVE ROUTE LOAD UP THROUGH NODE I.          00001330
C
C      CUMLN(I)   - CUMULATIVE ROUTE LENGTH UP THROUGH NODE I.         00001340
C
C

```

C DEMAND(I)	- DEMAND AT NODE I.	00001430
C DEPOT(I)	- ARTIFICIAL DEPOT OF ROUTE I.	00001440
C DIST(I,J)	- SHORTEST DISTANCE BETWEEN NODES I AND J.	00001450
C DISTLM	- MAXIMUM ROUTE LENGTH ALLOWED IN PROBLEM.	00001460
C DLIMIT	- LIMIT ON TOTAL DISTANCE IN SOLVING A ROUTE-LOAD DEVIATION OR ROUTE-LENGTH DEVIATION PROBLEM.	00001470 00001480
C DSEED	- DOUBLE PRECISION SEED USED IN CALL TO IMSL ROUTINES.	00001490
C DSTRSX	- GGUD AND GGUBFS.	00001500
C D1	- DISTANCE RELAXATION APPLIED TO TOTAL DISTANCE OBJECTIVE, USED IN GENERATING MULTIPLE FEASIBLE STARTING SOLUTIONS TO THE THE TOTAL DISTANCE PROBLEM.	00001510 00001520 00001530
C END	- DISTANCE BETWEEN POINT1 NODE AND ITS IMMEDIATE SUCCESSOR. SIMILAR DEFINITIONS APPLY TO D2 AND D3.	00001540 00001550
C EUCLID	- LAST NODE IN VRP NETWORK, OR LAST NODE IN TSP PROBLEM.	00001560
C FEAS	- O,1 VARIABLE INDICATING WHETHER EUCLIDEAN DISTANCES ARE TO BE USED IN PROBLEM. (0=NO, 1=YES).	00001570 00001580
C FEASLD(I)	- O,1 VARIABLE INDICATING WHETHER THE ROUTES RESULTING FROM AN ARC EXCHANGE ARE FEASIBLE WITH RESPECT TO PROBLEM CONSTRAINTS AND CONSTRAINING GOAL LEVELS.	00001590 00001600 00001610
C FEASLN(I)	- THE POTENTIAL LOAD OF ROUTE I RESULTING FROM AN ARC EXCHANGE. USED IN DETERMINING FEASIBILITY OF THE EXCHANGE.	00001620 00001630 00001640
C FEND	- THE POTENTIAL LENGTH OF ROUTE I RESULTING FROM AN ARC EXCHANGE. USED IN DETERMINING FEASIBILITY OF THE EXCHANGE.	00001650 00001660 00001670
C FPRED(I)	- THE POTENTIAL PREDECESSOR OF NODE I RESULTING FROM AN ARC EXCHANGE. USED IN DETERMINING FEASIBILITY OF THE EXCHANGE.	00001680 00001690
C FSTART	- THE POTENTIAL START OF A ROUTE BEING EXAMINED FOR FEASIBILITY OF THE EXCHANGE.	00001700 00001710 00001720
C FSUCC(I)	- THE POTENTIAL SUCCESSOR OF NODE I RESULTING FROM AN ARC EXCHANGE. USED IN DETERMINING FEASIBILITY OF THE EXCHANGE.	00001730 00001740
C HEAD(I)	- BEGINNING NODE IN ROUTE I FORMED BY THE CLARKE AND WRIGHT SAVINGS ALGORITHM.	00001750 00001760 00001770
C IACTRT	- THE NUMBER OF ACTIVE ROUTES IN PROBLEM; I.E., ROUTES HAVING AT LEAST ONE CUSTOMER.	00001780 00001790
C IANS	- RESPONSE FROM TERMINAL QUERY.	00001800
C IFLAG(I)	- AN INDICATOR THAT THE ITH VALUE IN THE SAVINGS FILE HAS BEEN MOVED. USED IN ALTERING THE SAVINGS FILE FOR RANDOM STARTING SOLUTIONS.	00001810 00001820 00001830 00001840 00001850
C IR(I)	- VECTOR OF SAVINGS VALUES, USED IN ALTERING THE SAVINGS FILE FOR RANDOM STARTING SOLUTIONS.	00001860 00001870
C IROUTE	- NUMBER OF ROUTES IN SOLUTION.	00001880
C IRTN	- STATEMENT NUMBER (BY ASSIGNMENT).	00001890
C ISORT(I)	- SORTED VALUE OF BEGINNING NODE (I) ASSOCIATED WITH SAVINGS LINK (I,J).	00001900 00001910
C ITRADE(,)	- VECTOR OF NONDOMINATED TRADEOFFS. SEE TRADE(,).	00001920
C JSORT(I)	- SORTED VALUE OF ENDING NODE (J) ASSOCIATED WITH SAVINGS LINK (I,J).	00001930 00001940
C LDDEV	- CURRENT VALUE OF ROUTE LOAD DEVIATION.	00001950
C LDDVLM	- LIMIT ON ROUTE LOAD DEVIATION IN SOLVING A TOTAL DISTANCE OR ROUTE LENGTH DEVIATION PROBLEM.	00001960 00001970
C LDRXLX	- AMOUNT OF RELAXATION APPLIED TO LDDEV IN FIRST ITERATION OF ROUTE LOAD DEVIATION ALGORITHM. USED FOR FINDING ALTERNATE STARTING SOLUTIONS.	00001980 00001990 00002000
C LENGTH(I)	- DISTANCE TRAVELED OVER ROUTE I.	00002010
C LIMIT1	- AMOUNT OF FIRST CONSTRAINING ACHIEVEMENT LEVEL. SEE CNSTR1.	00002020 00002030
C LIMIT2	- AMOUNT OF SECOND CONSTRAINING ACHIEVEMENT LEVEL. SEE CNSTR2.	00002040 00002050
C LOAD(I)	- TOTAL DEMAND CARRIED BY THE ITH VEHICLE.	00002060
C LOKK(I)	- O,1 VARIABLE DENOTING WHETHER A NODE IS TO BE INCLUDED IN CALCULATING LNDEV AND LDDEV. (0=YES, 1=NO).	00002070 00002080
C LNDEV	- CURRENT VALUE OF ROUTE-LENGTH DEVIATION.	00002090
C LNDVLM	- LIMIT ON ROUTE-LENGTH DEVIATION IN SOLVING A TOTAL DISTANCE OR ROUTE-LOAD DEVIATION PROBLEM.	00002100 00002110
C LNRLX	- AMOUNT OF RELAXATION APPLIED TO LNDEV IN FIRST ITERATION OF ROUTE-LENGTH DEVIATION ALGORITHM. USED	00002120 00002130

C	MAXLD	FOR FINDING ALTERNATE STARTING SOLUTIONS.	00002140
C	MAXLN	- AMOUNT OF DEMAND CARRIED BY MOST HEAVILY LOADED VEHICLE.	00002150
C	MAXLN	- DISTANCE DRIVEN OVER LONGEST ROUTE.	00002160
C	MENU	- MENU OPTION NUMBER SELECTED.	00002170
C	MINLD	- AMOUNT OF DEMAND CARRIED BY LEAST HEAVILY LOADED VEHICLE.	00002180
C	MINLN	- DISTANCE DRIVEN OVER SHORTEST ROUTE.	00002190
C	NCITY	- NUMBER OF CUSTOMERS IN PROBLEM.	00002200
C	NT	- NUMBER OF NONDOMINATED (REDUCED) TRADEOFFS.	00002210
C	NTRADE	- NUMBER OF ORIGINAL TRADEOFFS.	00002220
C	NULL	- 0,1 VARIABLE INDICATING WHETHER AN EXCHANGE IS NULL (0=NO, 1=YES).	00002230
C	OBJ	- CHARACTER VARIABLE CONTAINING NAME OF CURRENT OBJECTIVE FUNCTION BEING MINIMIZED.	00002240
C	OLDOBJ	- NUMBER OF OBJECTIVE FUNCTION MINIMIZED IN PREVIOUS ITERATION OF PROCEDURE.	00002250
C	PERMI(I)	- VARIABLE TO HOLD BEGINNING NODE OF ITH SAVINGS LINK. USED TO RESTORE SAVINGS FILE BACK TO ITS ORIGINAL SORTED ORDER AFTER A RANDOM PERTURBATION OF THE FILE HAS BEEN MADE.	00002260
C	PERMJ(I)	- VARIABLE TO HOLD ENDING NODE OF ITH SAVINGS LINK. CORRESPONDS TO PERMI(I).	00002270
C	PERMSV(I)	- VARIABLE TO HOLD THE ITH SAVINGS VALUE. USED TO RESTORE SAVINGS FILE TO ITS ORIGINAL SORTED ORDER AFTER A RANDOM PERTURBATION OF THE FILE HAS BEEN MADE	00002280
C	PNAME	- CHARACTER VARIABLE CONTAINING PROBLEM IDENTIFICATION.	00002290
C	POINT1	- BEGINNING NODE OF FIRST ARC TO BE REPLACED IN AN ARC EXCHANGE.	00002300
C	POINT2	- BEGINNING NODE OF SECOND ARC TO BE REPLACED IN AN ARC EXCHANGE.	00002310
C	POINT3	- BEGINNING NODE OF THIRD ARC TO BE REPLACED IN AN ARC EXCHANGE.	00002320
C	PRED(I)	- PREDECESSOR OF THE ITH NODE.	00002330
C	RTSIZE(I)	- NUMBER OF CUSTOMERS SERVED ON THE ITH ROUTE.	00002340
C	SAVING(I,J)	-SAVINGS CRITERION FOR LINKING NODES I AND J.	00002350
C	SOLNO	- SOLUTION NUMBER.	00002360
C	SORT(I)	- ITH SORTED SAVINGS VALUE.	00002370
C	START	- BEGINNING NODE OF A TSP, OR BEGINNING NODE OF AN ARC EXCHANGE ALGORITHM.	00002380
C	STCULD(I,J)	-VALUE OF CUMLD(J) STORED FOR THE ITH SOLUTION.	00002390
C	STCULN(I,J)	-VALUE OF CUMLN(J) STORED FOR THE ITH SOLUTION.	00002400
C	STDLIM	- VALUE OF DLIMIT STORED FOR THE ITH SOLUTION.	00002410
C	STDEP(I,J)	- VALUE OF DEPOT(J) STORED FOR THE ITH SOLUTION.	00002420
C	STITRD(I,,)	-VALUE OF ITRADE(,,) STORED FOR THE ITH SOLUTION.	00002430
C	STLDVL(I)	- VALUE OF LDDVLM STORED FOR THE ITH SOLUTION.	00002440
C	STLNTH(I,J)	-VALUE OF LENGTH(J) STORED FOR THE ITH SOLUTION.	00002450
C	STLNVL(I)	- VALUE OF LNDVLM STORED FOR THE ITH SOLUTION.	00002460
C	STLOAD(I,J)	-VALUE OF LOAD(J) STORED FOR THE ITH SOLUTION.	00002470
C	STLOKK(I,J)	-VALUE OF LOKK(J) STORED FOR THE ITH SOLUTION.	00002480
C	STMNLD(I)	- VALUE OF MINLD STORED FOR THE ITH SOLUTION.	00002490
C	STMNLN(I)	- VALUE OF MINLN STORED FOR THE ITH SOLUTION.	00002500
C	STMXLD(I)	- VALUE OF MAXLD STORED FOR THE ITH SOLUTION.	00002510
C	STMXLN(I)	- VALUE OF MAXLN STORED FOR THE ITH SOLUTION.	00002520
C	STNT(I)	- VALUE OF NT STORED FOR THE ITH SOLUTION.	00002530
C	STOBJ(I)	- VALUE OF OBJ STORED FOR THE ITH SOLUTION.	00002540
C	STPRED(I,J)	-VALUE OF PRED(J) STORED FOR THE ITH SOLUTION.	00002550
C	STSUCC(I,J)	-VALUE OF SUCC(J) STORED FOR THE ITH SOLUTION.	00002560
C	STTRK(I,J)	-VALUE OF TRUCK(J) STORED FOR THE ITH SOLUTION.	00002570
C	SUCC(I)	- SUCCESSOR OF THE ITH NODE.	00002580
C	TAIL(I)	- LAST NODE IN ROUTE I FORMED BY CLARKE AND WRIGHT SAVINGS ALGORITHM.	00002590
C	TARRAY(I)	- ARRAY FOR TERMINAL CONTROL VIA PLOT 10 PACKAGE.	00002600
C	TDIST	- TOTAL DISTANCE OVER ALL ROUTES.	00002610
C	TIME	- ELAPSED CPU TIME.	00002620
C	TRADE(,,)	- ARRAY OF ORIGINAL TRADEOFFS. SEE ITRADE(,,).	00002630
C	TRUCK(I)	- VEHICLE SERVING ITH CUSTOMER.	00002640
C	TRY(I,J)	- 0,1 VARIABLE INDICATING WHETHER A 2-ARC EXCHANGE HAS ALREADY BEEN EVALUATED FOR THE TWO ARCS BEGINNING WITH NODES I AND J. USED TO PREVENT MULTIPLE EVALUATIONS OF THE SAME EXCHANGE.	00002650

```

C      WK(I)      - WORK VECTOR USED IN CALL TO IMSL ROUTINE VSORA.      00002850
C      WORK(I)    - WORK VECTOR USED IN CALL TO IMSL ROUTINE VSORA.      00002860
C      WTLIM     - VEHICLE CAPACITY (WEIGHT LIMIT).                      00002870
C      XCENTR(I) - HORIZONTAL COMPONENT OF CENTROID OF ITH ROUTE.       00002880
C      XCOORD(I) - HORIZONTAL COORDINATE OF NODE I.                      00002890
C      YCENTR(I) - VERTICAL COMPONENT OF CENTROID OF ITH ROUTE.        00002900
C      YCOORD(I) - VERTICAL COORDINATE OF NODE I.                      00002910
C                                         00002920
C                                         00002930
C*****00002940
C                                         00002950
C      MAIN PROGRAM                                         00002960
C                                         00002970
C*****00002980
C                                         00002990
C                                         00003000
C      CHARACTER*44 PNAME,IPPLACE                                00003010
C      CHARACTER*16 DBJ,STOBJ(12),CNSTR1,CNSTR2                  00003020
C      CHARACTER*1 ANSWER                                     00003030
C      INTEGER EUCLID,CITY,XCOORD(0:120),YCOORD(0:120),DEMAND(0:120) 00003040
C      INTEGER HEAD(120),TAIL(120),PRED(120),SUCC(120),ROUTES,TWGT,WTLIM 00003050
C      INTEGER DIST,ALLOW,TDIST,DISTLM,TRUCK,IR(100),IFLAG(40),      00003060
*     PERMI(40),PERMJ(40)                                     00003070
C      DOUBLE PRECISION DSEED                                00003080
C      INTEGER START,END,POINT1,POINT2,D,FEASLD(20),FEASLN(20)   00003090
C      INTEGER FTRUCK,FWD,BACK,TEMTRK(120),CUMLD(120),CUMLN(120) 00003100
C      INTEGER FOUND,TRCNT,TRK,TAG,D1,D2,D3,D4,D5,D6,D7,D8  00003110
C      INTEGER FSTART,FEND,FPRED(120),FSUCC(120),BACTRT,DSTRXL 00003120
C      INTEGER PERMPR(120),PERMSU(120),PERMTR(120),RTSIZE(20)  00003130
C      INTEGER BESTRT,BESTDS,BCUMLN(120),BCUMLD(120),BESTP(120) 00003140
C      INTEGER BESTS(120),BESTLN(20),BESTLD(20),BDEPOT(20),BESTTR(120) 00003150
C      INTEGER DLIMIT,TARRAY(40),DEPOT(20),CLRNDX(12),BLDDEV,BLNDEV 00003160
C      INTEGER BMAXLD,BMINLD,BMAXLN,BMINLN,LOKK(120),STLOKK(12,120) 00003170
C      INTEGER BNTRAD,I TRADE(7,500),OLDOBJ                   00003180
C      INTEGER SOLNO,STPRED(12,120),STSUCC(12,120),STTRK(12,120), 00003190
*     STCULN(12,120),STCULD(12,120),STDEP(12,20),STLOAD(12,20), 00003200
*     STLNTH(12,20),STMXLN(12),STMNLN(12),STMULD(12),STMNLD(12), 00003210
*     STITRD(12,7,500),STLDVL(12),STLNVL(12),STDLIM(12),STNT(12) 00003220
C      DIMENSION BTRADE(7,500),TRADE(7,500),WK(14)           00003230
C      DIMENSION DIST(0:120,0:120),SAVING(3,6000),SORT(6000),PERMSY(40) 00003240
C      DIMENSION ISORT(6000),JSORT(6000),LOAD(120),TRUCK(120)  00003250
C      DIMENSION LENGTH(120),WORK(6),XCENTR(20),YCENTR(20)     00003260
C      COMMON IROUTE,NCITY,IRUN,START,END,WTLIM,DISTLM,ALLOW,NPERM, 00003270
*     ICOUNT,LDDVLM,LNDVLM,DLIMIT,MAXLD,MINLD,MAXLN,MINLN,D1,D2,D3,D4, 00003280
*     D5,D6,DEPOT,PRED,SUCC,TRUCK,DEMAND,LENGTH,LOAD,CUMLD,CUMLN,TEMTRK, 00003290
*     FEASLD,FEASLN,PERMPR,PERMSU,PERMTR,ISORT,JSORT,SORT,DIST,XCOORD, 00003300
*     YCOORD,LOKK,TRADE,NTRADE,DSEED                         00003310
C      COMMON /STORE/STPRED,STSUCC,STTRK,STCULN,STCULD,STDEP,STLOAD, 00003320
*     STLNTH,STMXLN,STMNLN,STMULD,STMNLD,STITRD,STNT,STLDVL,STLNVL, 00003330
*     STDLIM,STLOKK,STOBJ                                    00003340
C                                         00003350
C                                         00003360
C                                         00003370
C                                         00003380
C                                         00003390
C      INITIALIZE TIMER                                     00003400
C                                         00003410
C                                         00003420
C
C      TIME=0.0                                              00003430
C      NTRADE=0                                              00003440
C      NT=0                                                 00003450
C      CALL ELAPSE(ITIME)                                 00003460
C      CALL ELAPSE(ITIME)                                 00003470
C      LDDVLM=99000                                         00003480
C      LNDVLM=99000                                         00003490
C                                         00003500
C                                         00003510
C                                         00003520
C                                         00003530
C      SET UP THE GRAPHICS SCREEN                         00003540
C                                         00003550

```

```

C          00003560
C          00003570
C          00003580
C          00003590
C          00003600
C          00003610
C          00003620
C          00003630
C          00003640
C          00003650
C          00003660
C          00003670
C          00003680
C          00003690
C          00003700
C          00003710
C          00003720
C          00003730
C          00003740
C          00003750
C          00003760
C          00003770
C          00003780
C          00003790
C          00003800
C          00003810
C          00003820
C          00003830
C          00003840
C          00003850
C          00003860
C          00003870
C          00003880
C          00003890
C          00003900
C          00003910
C          00003920
C          00003930
C          00003940
C          00003950
C          00003960
C          00003970
C          00003980
C          00003990
C          00004000
C          00004010
C          00004020
C          00004030
C          00004040
1 CONTINUE      00004050
LIM=MAXO(MAXX-MINX,MAXY-MINY)      00004060
X1=MINX-10      00004070
X2=X1+FLOAT(LIM)+20.      00004080
Y1=MINY-10      00004090
Y2=Y1+FLOAT(LIM)+20.      00004100
CALL DWINDO(X1,X2,Y1,Y2)      00004110
C          00004120
C          00004130
C          00004140
C          00004150
C          00004160
IF(EUCLID.EQ.1) THEN      00004170
DO 2 I=0,NCITY-1      00004180
DO 2 J=I+1,NCITY      00004190
DIST(I,J)=SQRT(FLOAT((XCOORD(I)-XCOORD(J))**2+(YCOORD(I)
*           - YCOORD(J))**2)) + 0.5      00004200
00004210
2 DIST(J,I)=DIST(I,J)      00004220
GOTO 4      00004230
END IF      00004240
C          00004250
C          00004260

```

```

C           IF PROBLEM HAS NON-EUCLIDEAN DISTANCES, READ THEM IN.      00004270
C
C
C           IF(EUCLID.EQ.0) THEN                                         00004280
3           READ(8,103,END=4) I,J,DIST(I,J)                           00004290
              DIST(J,I)=DIST(I,J)
              GOTO 3
END IF
4           CONTINUE
C
C           CALCULATE THE SAVINGS FILE.                                00004300
C
C
C           DSEED=6.0D0
C           ICOUNT=0
DO 6   I=1,NCITY-1
DO 6   J=I+1,NCITY
              ICOUNT=ICOUNT+1
              SAVING(1,ICOUNT)=FLOAT(DIST(0,I)+DIST(J,0)-DIST(I,J))*(-1.)
              SAVING(2,ICOUNT)=I
6           SAVING(3,ICOUNT)=J
C
C           SORT THE SAVINGS FILE -- IMSL MODIFIED QUICKSORT SUBROUTINE 00004310
C
C
C           CALL VSORA(SAVING,3,3,ICOUNT,1,WORK,IER)                   00004320
DO 7   I=1,ICOUNT
              SORT(I)=-SAVING(1,I)
              ISORT(I)=SAVING(2,I)
7           JSORT(I)=SAVING(3,I)
DO 8   I=1,40
              PERMSV(I)=SORT(I)
              PERMI(I)=ISORT(I)
8           PERMJ(I)=JSORT(I)
C
C           DO 9   I=1,120
9           LOKK(I)=0
DO 10  I=1,7
DO 10  J=1,500
10          TRADE(I,J)=0.
C
C           DO 70 IRUN=1,8
C
C           ALTER THE SAVINGS FILE FOR A NEW INITIAL RANDOM SOLUTION 00004500
C
C
C           IF(IRUN.GT.1) THEN                                         00004510
              CALL GGUD(DSEED,IRROUTE*2,2*(IRROUTE*2),IR)
DO 12  I=1,40
              IFLAG(I)=0
12          MOVES=0
DO 13  I=1,2*(IRROUTE*2)
              IF(IFLAG(IR(I)).GT.0) GOTO 13
C           ELSE
              MOVES=MOVES+1
              SORT(MOVES)=PERMSV(IR(I))
              ISORT(MOVES)=PERMI(IR(I))
              JSORT(MOVES)=PERMJ(IR(I))
              IFLAG(IR(I))=1
              IF(MOVES.GE.IROUTE*2) GOTO 14
13          CONTINUE
END IF
14          CONTINUE

```

```

C          00004980
C          00004990
C          00005000
C          00005010
C          00005020
C          00005030
C          00005040
C          00005050
C          00005060
C          00005070
C          00005080
C          00005090
C          00005100
C          00005110
C          00005120
C          00005130
C          00005140
C          00005150
C          00005160
C          00005170
C          00005180
C          00005190
C          00005200
C          00005210
C          00005220
C          00005230
C          00005240
C          00005250
C          00005260
C          00005270
C          00005280
C          00005290
C          00005300
C          00005310
C          00005320
C          00005330
C          00005340
C          00005350
C          00005360
C          00005370
C          00005380
C          00005390
C          00005400
C          00005410
C          00005420
C          00005430
C          00005440
C          00005450
C          00005460
C          00005470
C          00005480
C          00005490
C          00005500
C          00005510
C          00005520
C          00005530
C          00005540
C          00005550
C          00005560
C          00005570
C          00005580
C          00005590
C          00005600
C          00005610
C          00005620
C          00005630
C          00005640
C          00005650
C          00005660
C          00005670
C          00005680

C
C          CALL SAVNGS
C
C          RESTORE SAVINGS FILE TO ORIGINAL SORTED ORDER
C
C          DO 15 I=1,40
C              SORT(I)=PERMSV(I)
C              ISORT(I)=PERMI(I)
C          15    JSORT(I)=PERMJ(I)
C
C          29  FWD=1
C              BACK=2
C              START=NCITY+1
C              END=PRED(START)
C              INSIDE=0
C              NEXT=NCITY+1
C          31  NODE=NEXT
C              IF(NODE.EQ.NCITY+1.AND.INSIDE.EQ.1) GOTO 32
C              INSIDE=1
C              IF(NODE.GT.NCITY) THEN
C                  ILN=0
C                  ILD=0
C                  CUMLD(NODE)=0
C                  CUMLN(NODE)=0
C              ENDIF
C              ILD=ILD+DEMAND(NODE)
C              CUMLD(NODE)=ILD
C              IF(NODE.LE.NCITY) ILN=ILN+DIST(NODE,PRED(NODE))+ALLOW
C              CUMLN(NODE)=ILN
C              NEXT=SUCC(NODE)
C              GOTO 31
C          32  CONTINUE
C              CALL TWOOPT(0)
C              CALL THROPT(0)
C              LOADT=0
C              LENGTH=0
C              DO 33 I=1,IROUTE
C                  LOADT=LOADT+LOAD(I)
C          33    LENGTH=LENGTH+LENGTH(I)
C
C          SAVE BEST SOLUTION.
C
C          IF(IRUN.EQ.1) THEN
C              IACTRT=0
C              DO 59 I=1,IROUTE
C                  IF(LOAD(I).GT.0) IACTRT=IACTR+1
C          59    CONTINUE
C              BACTRT=IACTR
C              BESTRT=IRROUTE
C              BESTDS=LENGTH
C              DO 63 I=1,NCITY+IRROUTE
C                  BESTP(I)=PRED(I)
C                  BESTS(I)=SUCC(I)
C                  BESTTR(I)=TRUCK(I)
C                  BCUMLN(I)=CUMLN(I)
C          63    BCUMLD(I)=CUMLD(I)
C              DO 64 I=1,IROUTE
C                  BESTLD(I)=LOAD(I)
C                  BESTLN(I)=LENGTH(I)
C          64    BDEPOT(I)=DEPOT(I)
C              GOTO 70
C          ENDIF

```

```

IACTR=0          00005690
DO 65 I=1,IROUTE 00005700
  IF(LOAD(I).GT.0) IACTR=IACTR+1 00005710
65 CONTINUE      00005720
  IF(IACTR.GT.BACTR) GOTO 70 00005730
  IF(IACTR.EQ.BACTR.AND.LENGT.GE.BESTDS) GOTO 70 00005740
C   ELSE          00005750
    BESTRT=IROUTE 00005760
    BESTDS=LENGT 00005770
    BACTR=IACTR 00005780
    DO 66 I=1,NCITY+IROUTE 00005790
      BESTP(I)=PRED(I) 00005800
      BESTS(I)=SUCC(I) 00005810
      BESTTR(I)=TRUCK(I) 00005820
      BCUMLN(I)=CUMLN(I) 00005830
66   BCUMLD(I)=CUMLD(I) 00005840
    DO 67 I=1,IROUTE 00005850
      BESTLD(I)=LOAD(I) 00005860
      BESTLN(I)=LENGTH(I) 00005870
67   BDEPOT(I)=DEPOT(I) 00005880
70 CONTINUE      00005890
C
C
C   RECALL BEST SOLUTION. 00005900
C
C
  IROUTE=BESTRT 00005910
  LENGTH=BESTDS 00005920
  DO 71 I=1,NCITY+IROUTE 00005930
    PRED(I)=BESTP(I) 00005940
    SUCC(I)=BESTS(I) 00005950
    CUMLN(I)=BCUMLN(I) 00005960
    CUMLD(I)=BCUMLD(I) 00005970
71   TRUCK(I)=BESTTR(I) 00005980
    TDIST=0          00005990
    DO 72 I=1,IROUTE 00006000
      RTSIZE(I)=0.0 00006010
      XCENTR(I)=0.0 00006020
      YCENTR(I)=0.0 00006030
      LOAD(I)=BESTLD(I) 00006040
      LENGTH(I)=BESTLN(I) 00006050
      TDIST=TDIST+LENGTH(I) 00006060
72   DEPOT(I)=BDEPOT(I) 00006070
    CALL TWINDO(1010,4095,0,2800) 00006080
    CALL CHRISZ(3) 00006090
    NEXT=NCITY+1 00006100
    X=XCOORD(NCITY+1) 00006110
    Y=YCOORD(NCITY+1) 00006120
    XCENTR(TRUCK(NCITY+1))=XCENTR(TRUCK(NCITY+1)) + X 00006130
    YCENTR(TRUCK(NCITY+1))=YCENTR(TRUCK(NCITY+1)) + Y 00006140
    RTSIZE(TRUCK(NCITY+1))=RTSIZE(TRUCK(NCITY+1)) + 1 00006150
    CALL MOVEA(X,Y) 00006160
    CALL MOVREL(40,0) 00006170
    CALL DRWREL(0,40) 00006180
    CALL DRWREL(-80,0) 00006190
    CALL DRWREL(0,-80) 00006200
    CALL DRWREL(80,0) 00006210
    CALL DRWREL(0,40) 00006220
    CALL MOVEA(X,Y) 00006230
    INSIDE=0          00006240
73 NODE=NEXT      00006250
  IF(NODE.EQ.NCITY+1.AND.INSIDE.EQ.1) GOTO 74 00006260
  INSIDE=1          00006270
  X=XCOORD(NODE) 00006280
  Y=YCOORD(NODE) 00006290
  XCENTR(TRUCK(NODE))=XCENTR(TRUCK(NODE)) + X 00006300
  YCENTR(TRUCK(NODE))=YCENTR(TRUCK(NODE)) + Y 00006310
  RTSIZE(TRUCK(NODE))=RTSIZE(TRUCK(NODE)) + 1 00006320
  CALL DRAWA(X,Y) 00006330
  ICHR1=NODE/100 00006340

```

```

ICHR2=NODE/10 - ICHR1*10          00006400
ICHR3=NODE-(ICHR1*100 + ICHR2*10) 00006410
IF(NODE.LE.NCITY) THEN            00006420
  CALL MOVREL(20,0)                00006430
  CALL DRWREL(0,20)                00006440
  CALL DRWREL(-40,0)               00006450
  CALL DRWREL(0,-40)               00006460
  CALL DRWREL(40,0)                00006470
  CALL DRWREL(0,20)                00006480
ENDIF
CALL MOVEA(X,Y)                  00006490
NEXT=SUCC(NODE)                  00006500
GOTO 73                          00006520
74 X=XCOORD(NODE)                00006530
Y=YCOORD(NODE)                  00006540
CALL DRAWA(X,Y)                  00006550
CALL CHRISIZ(1)                  00006560
DO 744 J=1,IROUTE                00006570
  IF(LOAD(J).GT.0) THEN           00006580
    XCENTR(J)=XCENTR(J)/FLOAT(RTSIZE(J)) 00006590
    YCENTR(J)=YCENTR(J)/FLOAT(RTSIZE(J)) 00006600
    CALL MOVEA(XCENTR(J),YCENTR(J))      00006610
    ICHR1=J/10                     00006620
    ICHR2=J-ICHR1*10               00006630
    IF(ICHR1.NE.0) CALL ANCHO(ICHR1+48) 00006640
    CALL ANCHO(ICHR2+48)           00006650
  ENDIF
744 CONTINUE                      00006660
  CALL CHRISIZ(1)                 00006670
  IMAXLN=-99                      00006680
  IMAXLD=-99                      00006690
  IMINLN=999999                   00006700
  IMINLD=999999                   00006710
  DO 75 I=1,IROUTE                00006720
    IF(LOKK(DEPOT(I)).NE.0) GOTO 75
    IF(LENGTH(I).GT.IMAXLN) IMAXLN=LENGTH(I) 00006730
    IF(LENGTH(I).LT.IMINLN.AND.LENGTH(I).GT.0) IMINLN=LENGTH(I) 00006740
    IF(LOAD(I).GT.IMAXLD) IMAXLD=LOAD(I) 00006750
    IF(LOAD(I).LT.IMINLD.AND LOAD(I).GT.0) IMINLD=LOAD(I) 00006760
  75 CONTINUE                      00006770
  MAXLN=IMAXLN                    00006780
  MINLN=IMINLN                    00006790
  MAXLD=IMAXLD                    00006800
  MINLD=IMINLD                    00006810
  LDDEV=MAXLD-MINLD               00006820
  LNDEV=MAXLN-MINLN               00006830
  C                                00006840
  C                                00006850
  C                                00006860
  C                                00006870
  C                                00006880
  C WRITE OUT THE BEGINNING ROUTE SET. 00006890
  C                                00006900
  C                                00006910
  SOLNO=1                         00006920
  CALL MOVABS(1030,2950)           00006930
  CALL AOUTST(44,PNAME)            00006940
  CALL TWINDO(0,4095,0,3120)       00006950
  CALL HOME                         00006960
  CALL ANMODE                        00006970
  WRITE(6,319)                      00006980
319 FORMAT(1H ,//' SOLUTION NUMBER 1'//' BEGINNING ROUTE SET'
          *//' ROUTE LOAD LENGTH//')
  DO 76 II=1,IROUTE                00006990
    IF(LOAD(II).GT.0) WRITE(6,321) II,LOAD(II),LENGTH(II) 00007000
  76 CONTINUE                      00007010
321 FORMAT(I4,I8,1X,I5)            00007020
  WRITE(6,322) TDIST,LDDEV,LNDEV   00007030
322 FORMAT(' TOT. DIST =',I6,',',LOAD DEV. = ',I5,',',LENGTH DEV. = ',I5)
          *4)                           00007040
  CALL ELAPSE(ITIME)                00007050
  TIME=TIME+FLOAT(ITIME)/1000.      00007060
  WRITE(6,323) TIME                 00007070
                                            00007080
                                            00007090
                                            00007100

```

```

323 FORMAT(1H , 'CPU SECONDS:', F6.2)          00007110
      WRITE(6,400)                                00007120
      READ(5,100) ANSWER                         00007130
      IF(ANSWER.EQ.'Y') THEN                      00007140
          CALL FINITT(0,700)                       00007150
          STOP                                     00007160
      ENDIF                                     00007170
C
C
C
C      STORE THE BEGINNING SOLUTION             00007180
C
C
C      STDLIM(SOLNO)=DLIMIT                     00007190
C      STLDVL(SOLNO)=LDDVLM                     00007200
C      STLNVL(SOLNO)=LNDVLM                     00007210
C      STMNLD(SOLNO)=MINLD                      00007220
C      STMXLD(SOLNO)=MAXLD                      00007230
C      STMNLN(SOLNO)=MINLN                      00007240
C      STMXLN(SOLNO)=MAXLN                      00007250
C      STNT(SOLNO)=0                            00007260
C      STOBJ(SOLNO)='TOTAL DISTANCE'            00007270
C      DO 78 J=1,IROUTE                         00007280
C          STDEP(SOLNO,J)=DEPOT(J)              00007290
C          STLNTH(SOLNO,J)=LENGTH(J)            00007300
78      STLOAD(SOLNO,J)=LOAD(J)                 00007310
C      DO 79 J=1,NCITY+IROUTE                   00007320
C          STCULD(SOLNO,J)=CUMLD(J)             00007330
C          STCULN(SOLNO,J)=CUMLN(J)             00007340
C          STLOKK(SOLNO,J)=LOKK(J)               00007350
C          STPRED(SOLNO,J)=PRED(J)               00007360
C          STSUCC(SOLNO,J)=SUCC(J)              00007370
C      79      STTRK(SOLNO,J)=TRUCK(J)           00007380
C
C
C      ELSE
C          WRITE(6,402)
C          READ(5,*) IANS
C
C
C      SOLVE A ROUTE-LENGTH DEVIATION PROBLEM   00007390
C
C      IF(IANS.EQ.1) THEN                        00007400
C          OBJ='LENGTH DEVIATION'                00007410
C          OLDOBJ=3                           00007420
C          DLIMIT=TDIST                      00007430
C          LDDVLM=LDDEV                      00007440
C          CNSTR1='TOTAL DISTANCE'            00007450
C          CNSTR2='LOAD DEVIATION'            00007460
C          CALL NEWPAG                         00007470
C          WRITE(6,404) OBJ,CNSTR1,DLIMIT,CNSTR2,LDDVLM 00007480
C          LIMIT1=DLIMIT                     00007490
C          LIMIT2=LDDVLM                     00007500
C          NT=0                               00007510
C          CALL LNDV2(0)                      00007520
C          CALL LNDV3(0)                      00007530
C          BNTRAD=NTRADE                     00007540
C          DO 77 I=1,NTRADE                  00007550
C          DO 77 J=1,7                         00007560
77      BTRADE(J,I)=TRADE(J,I)                 00007570
C          DO 80 I=1,IROUTE                  00007580
C              BESTLD(I)=LOAD(I)              00007590
C              BESTLN(I)=LENGTH(I)            00007600
80      BDEPOT(I)=DEPOT(I)                   00007610
C          DO 81 I=1,NCITY+IROUTE            00007620
C              BESTP(I)=PRED(I)              00007630
C              BESTS(I)=SUCC(I)              00007640
C              BESTTR(I)=TRUCK(I)            00007650
C              BCUMLN(I)=CUMLN(I)            00007660
C      81      BCUMLD(I)=CUMLD(I)              00007670
C          LNDEV=MAXLN-MINLN                00007680

```

```

BMAXLD=MAXLD          00007820
BMINLD=MINLD          00007830
BMAXLN=MAXLN          00007840
BMINLN=MINLN          00007850
BLNDEV=LNDEV          00007860
LNRLX=FLOAT(LNDEV)*0.5 00007870
CALL LNDV3(LNRLX)      00007880
LNDEV=MAXLN-MINLN     00007890
IF(LNDEV.GE.BLNDEV) THEN 00007900
  LNDEV=BLNDEV          00007910
  MAXLD=BMAXLD          00007920
  MINLD=BMINLD          00007930
  MAXLN=BMAXLN          00007940
  MINLN=BMINLN          00007950
  DO 83 I=1,IRROUTE    00007960
    LOAD(I)=BESTLD(I)    00007970
    LENGTH(I)=BESTLN(I)  00007980
  83   DEPOT(I)=BDEPOT(I) 00007990
  DO 84 I=1,NCITY+IRROUTE 00008000
    PRED(I)=BESTP(I)    00008010
    SUCC(I)=BESTS(I)    00008020
    TRUCK(I)=BESTTR(I)  00008030
    CUMLN(I)=BCUMLN(I)  00008040
  84   CUMLD(I)=BCUMLD(I) 00008050
  NTRADE=BNTRAD         00008060
  DO 82 I=1,NTRADE      00008070
  DO 82 J=1,7            00008080
  82   TRADE(J,I)=BTRADE(J,I) 00008090
  ENDIF
  IF(NTRADE.GT.0) CALL VSORA(TRADE,7,7,NTRADE,1,WK,IER) 00008110
  NT=0                  00008120
  IF(NTRADE.EQ.0) GOTO 93 00008130
C  ELSE                  00008140
  PREV1=999999.          00008150
  PREV2=-999999.         00008160
  PREV3=-999999.         00008170
  DO 823 I=1,NTRADE     00008180
    IF(TRADE(1,I).NE.PREV1) THEN 00008190
      NT=NT+1              00008200
      ITRADE(1,NT)=TRADE(1,I) 00008210
      ITRADE(2,NT)=TRADE(2,I) 00008220
      ITRADE(3,NT)=TRADE(3,I) 00008230
      ITRADE(4,NT)=TRADE(4,I) 00008240
      ITRADE(5,NT)=TRADE(5,I) 00008250
      ITRADE(6,NT)=TRADE(6,I) 00008260
      ITRADE(7,NT)=TRADE(7,I) 00008270
      PREV1=TRADE(1,I)       00008280
      PREV2=TRADE(2,I)       00008290
      PREV3=TRADE(3,I)       00008300
      GOTO 823              00008310
    ENDIF
    IF(TRADE(2,I).LE.PREV2.AND.TRADE(3,I).LE.PREV3) THEN 00008330
      ITRADE(2,NT)=TRADE(2,I) 00008340
      ITRADE(3,NT)=TRADE(3,I) 00008350
      ITRADE(4,NT)=TRADE(4,I) 00008360
      ITRADE(5,NT)=TRADE(5,I) 00008370
      ITRADE(6,NT)=TRADE(6,I) 00008380
      ITRADE(7,NT)=TRADE(7,I) 00008390
      PREV1=TRADE(1,I)       00008400
      PREV2=TRADE(2,I)       00008410
      PREV3=TRADE(3,I)       00008420
    ENDIF
  823  CONTINUE           00008430
  CALL NONDOM(ITRADE,NT) 00008440
  GOTO 93                00008450
  ENDIF
C
C   SOLVE A ROUTE-LOAD DEVIATION PROBLEM. 00008460
C                                         00008470
C                                         00008480
C                                         00008490
C                                         00008500
C                                         00008510
C                                         00008520

```

```

IF(IANS.EQ.2) THEN          00008530
  OBJ='LOAD DEVIATION'
  OLDOBJ=2                  00008540
  DLIMIT=TDIST               00008550
  LNDVLM=LNDDEV              00008560
  CNSTR1='TOTAL DISTANCE'    00008570
  CNSTR2='LENGTH DEVIATION'  00008580
  CALL NEWPAG                00008590
  WRITE(6,404) OBJ,CNSTR1,DLIMIT,CNSTR2,LNDVLM 00008600
  LIMIT1=DLIMIT               00008610
  LIMIT2=LNDVLM               00008620
  NT=0                         00008630
  DO 901 IRUN=1,3             00008640
    LDRLX=FLDAT(IRUN-1)*FLOAT(LDDEV)*0.5      00008650
    IF(IRUN.EQ.1) CALL LDDV2(0)                 00008660
    CALL LDDV3(LDRLX)                          00008670
    LDDEV=MAXLD-MINLD                      00008680
    IF(IRUN.EQ.1.OR.LDDEV.LT.BLDDEV) THEN     00008690
      BNTRAD=NTRADE
      DO 85 I=1,NTRADE                   00008700
        DO 85 J=1,7                     00008710
          BTRADE(J,I)=TRADE(J,I)           00008720
          DO 85 I=1,IROUTE                00008730
            BESTLD(I)=LOAD(I)             00008740
            BESTLN(I)=LENGTH(I)
            BDEPOT(I)=DEPOT(I)
            DO 88 I=1,NCITY+IROUTE       00008750
              BESTP(I)=PRED(I)
              BESTS(I)=SUCC(I)
              BESTTR(I)=TRUCK(I)
              BCUMLN(I)=CUMLN(I)
              BCUMLD(I)=CUMLD(I)
              BMAXLD=MAXLD
              BMINLD=MINLD
              BMAXLN=MAXLN
              BMINLN=MINLN
              BLDDEV=LDDEV
              ENDIF
            901  CONTINUE
              LDDEV=BLDDEV
              MAXLD=BMAXLD
              MINLD=BMINLD
              MAXLN=BMAXLN
              MINLN=BMINLN
              DO 89 I=1,IROUTE
                LOAD(I)=BESTLD(I)
                LENGTH(I)=BESTLN(I)
                BDEPOT(I)=BDEPOT(I)
                DO 90 I=1,NCITY+IROUTE
                  PRED(I)=BESTP(I)
                  SUCC(I)=BESTS(I)
                  TRUCK(I)=BESTTR(I)
                  CUMLN(I)=BCUMLN(I)
                  CUMLD(I)=BCUMLD(I)
                  NTRADE=BNTRAD
                  DO 91 I=1,NTRADE
                    DO 91 J=1,7
                      TRADE(J,I)=BTRADE(J,I)
                      IF(NTRADE.GT.0) CALL VSORA(TRADE,7,7,NTRADE,1,WK,IER)
                      NT=0
                      IF(NTRADE.EQ.0) GOTO 93
                      ELSE
                        PREV1=999999.
                        PREV2=-999999.
                        PREV3=-999999.
                        DO 826 I=1,NTRADE
                          IF(TRADE(1,I).NE.PREV1) THEN
                            NT=NT+1
                            ITRADE(1,NT)=TRADE(1,I)
                            ITRADE(2,NT)=TRADE(2,I)
                            ITRADE(3,NT)=TRADE(3,I)
                          ENDIF
                        ENDIF
                      ENDIF
                    ENDIF
                  ENDIF
                ENDIF
              ENDIF
            ENDIF
          ENDIF
        ENDIF
      ENDIF
    ENDIF
  ENDIF
ENDIF
  
```

```

ITRADE(4,NT)=TRADE(4,I)          00009240
ITRADE(5,NT)=TRADE(5,I)          00009250
ITRADE(6,NT)=TRADE(6,I)          00009260
ITRADE(7,NT)=TRADE(7,I)          00009270
PREV1=TRADE(1,I)                 00009280
PREV2=TRADE(2,I)                 00009290
PREV3=TRADE(3,I)                 00009300
GOTO 826                         00009310
ENDIF                           00009320
IF(TRADE(2,I).LE.PREV2.AND.TRADE(3,I).LE.PREV3) THEN 00009330
    ITRADE(2,NT)=TRADE(2,I)      00009340
    ITRADE(3,NT)=TRADE(3,I)      00009350
    ITRADE(4,NT)=TRADE(4,I)      00009360
    ITRADE(5,NT)=TRADE(5,I)      00009370
    ITRADE(6,NT)=TRADE(6,I)      00009380
    ITRADE(7,NT)=TRADE(7,I)      00009390
    PREV1=TRADE(1,I)             00009400
    PREV2=TRADE(2,I)             00009410
    PREV3=TRADE(3,I)             00009420
ENDIF                           00009430
826   CONTINUE                   00009440
CALL NDOMOM(ITRADE,NT)           00009450
GOTO 93                          00009460
ENDIF                           00009470
C                                00009480
C                                00009490
C                                00009500
C DISPLAY SOLUTION ON GRAPHICS SCREEN 00009510
C                                00009520
C                                00009530
C                                00009540
93  SOLNO=SOLNO+1                00009550
NEXT=NCITY+1                      00009560
CALL NEWPAG                        00009570
CALL TWINDO(0,4095,0,3120)         00009580
CALL MOVABS(0,0)                   00009590
CALL DRWABS(4095,0)                00009600
CALL DRWABS(4095,3120)             00009610
CALL DRWABS(0,3120)                00009620
CALL DRWABS(0,0)                   00009630
CALL MOVABS(1030,0)                00009640
CALL DRWABS(1030,3120)              00009650
CALL MOVABS(1030,2800)              00009660
CALL DRWABS(4095,2800)              00009670
CALL MOVABS(1130,2950)              00009680
CALL AOUTST(44,PNAME)              00009690
CALL TWINDO(1030,4095,0,2800)       00009700
X1=MINX-10                         00009710
X2=X1+FLOAT(LIM)+20.               00009720
Y1=MINY-10                         00009730
Y2=Y1+FLOAT(LIM)+20.               00009740
CALL DWINDO(X1,X2,Y1,Y2)            00009750
CALL CHRISIZ(3)                     00009760
DO 94 J=1,IRROUTE                  00009770
    RTSIZE(J)=0                      00009780
    XCENTR(J)=0.0                     00009790
94  YCENTR(J)=0.0                   00009800
X=XCOORD(NEXT)                     00009810
Y=YCOORD(NEXT)                     00009820
XCENTR(TRUCK(NEXT))=XCENTR(TRUCK(NEXT)) + X 00009830
YCENTR(TRUCK(NEXT))=YCENTR(TRUCK(NEXT)) + Y 00009840
RTSIZE(TRUCK(NEXT))=RTSIZE(TRUCK(NEXT)) + 1 00009850
CALL MOVEA(X,Y)                    00009860
CALL MOVREL(40,0)                  00009870
CALL DRWREL(0,40)                  00009880
CALL DRWREL(-80,0)                 00009890
CALL DRWREL(0,-80)                 00009900
CALL DRWREL(80,0)                  00009910
CALL DRWREL(0,40)                  00009920
CALL MOVEA(X,Y)                    00009930
INSIDE=0                           00009940

```

```

95 NODE=NEXT          00009950
  IF(NODE.EQ.NCITY+1.AND.INSIDE.EQ.1) GOTO 96      00009960
  INSIDE=1          00009970
  X=XCOORD(NODE)    00009980
  Y=YCOORD(NODE)    00009990
  XCENTR(TRUCK(NODE))=XCENTR(TRUCK(NODE)) + X      00010000
  YCENTR(TRUCK(NODE))=YCENTR(TRUCK(NODE)) + Y      00010010
  RTSIZE(TRUCK(NODE))=RTSIZE(TRUCK(NODE)) + 1      00010020
  CALL DRAWA(X,Y)    00010030
  ICHR1=NODE/100     00010040
  ICHR2=NODE/10 - ICHR1*10     00010050
  ICHR3=NODE - (ICHR1*100 + ICHR2*10)     00010060
  IF(NODE.LE.NCITY) THEN     00010070
    CALL MOVREL(20,0)        00010080
    CALL DRWREL(0,20)        00010090
    CALL DRWREL(-40,0)       00010100
    CALL DRWREL(0,-40)       00010110
    CALL DRWREL(40,0)        00010120
    CALL DRWREL(0,20)        00010130
  ENDIF                 00010140
  CALL MOVEA(X,Y)        00010150
  NEXT=SUCC(NODE)        00010160
  GOTO 95                 00010170
96 CONTINUE             00010180
  X=XCOORD(NODE)        00010190
  Y=YCOORD(NODE)        00010200
  CALL DRAWA(X,Y)        00010210
  CALL CHRSLZ(1)         00010220
  DO 97 J=1,IROUTE       00010230
    IF(LOAD(J).GT.0) THEN     00010240
      XCENTR(J)=XCENTR(J)/FLOAT(RTSIZE(J))      00010250
      YCENTR(J)=YCENTR(J)/FLOAT(RTSIZE(J))      00010260
      CALL MOVEA(XCENTR(J),YCENTR(J))      00010270
      ICHR1=J/10          00010280
      ICHR2=J-ICHR1*10     00010290
      IF(ICHR1.NE.0) CALL ANCHO(ICHR1+48)      00010300
      CALL ANCHO(ICHR2+48)      00010310
    ENDIF                 00010320
  97 CONTINUE             00010330
  CALL CHRSLZ(1)          00010340
  TDIST=0                 00010350
  DO 98 I=1,IROUTE       00010360
  98 TDIST=TDIST+LENGTH(I)      00010370
  LNDEV=MAXLN-MINLN        00010380
  LDDEV=MAXLD-MINLD        00010390
  CALL HOME                00010400
  CALL ANMODE               00010410
  WRITE(6,405) SOLNO,OBJ      00010420
  DO 99 II=1,IROUTE       00010430
  IF(LOAD(II).LE.0) GOTO 99      00010440
  WRITE(6,321) II,LOAD(II),LENGTH(II)      00010450
  99 CONTINUE             00010460
  WRITE(6,322) TDIST,LDDEV,LNDEV      00010470
  CALL ELAPSE(ITIME)        00010480
  TIME=TIME+FLOAT(ITIME)/1000.      00010490
  WRITE(6,323) TIME          00010500
  WRITE(6,450)              00010510
  450 FORMAT(1H ,//////////,' HIT <RTN> TO CONTINUE')
  CALL TINPUT(MMM)
C
C
C      STORE CURRENT SOLUTION
C
C      STDLIM(SOLNO)=DLIMIT      00010540
C      STLDVL(SOLNO)=LDDVLM      00010550
C      STLNVL(SOLNO)=LNDVLM      00010560
C      STMNLD(SOLNO)=MINLD      00010570
C      STMXLD(SOLNO)=MAXLD      00010580
C      STMNLN(SOLNO)=MINLN      00010590
C      STMXLN(SOLNO)=MAXLN      00010600

```

```

STNT(SOLNO)=NT          00010660
STOBJ(SOLNO)=OBJ        00010670
DO 130 J=1,IROUTE       00010680
  STDEP(SOLNO,J)=DEPOT(J) 00010690
  STLNTH(SOLNO,J)=LENGTH(J) 00010700
130  STLOAD(SOLNO,J)=LOAD(J) 00010710
  DO 140 J=1,NCITY+IROUTE 00010720
    STCULD(SOLNO,J)=CUMLD(J) 00010730
    STCULN(SOLNO,J)=CUMLN(J) 00010740
    STLOKK(SOLNO,J)=LOKK(J) 00010750
    STPRED(SOLNO,J)=PRED(J) 00010760
    STSUCC(SOLNO,J)=SUCC(J) 00010770
140  STTRK(SOLNO,J)=TRUCK(J) 00010780
  DO 145 I=1,7           00010790
    DO 145 J=1,NT          00010800
145  STITRD(SOLNO,I,J)=ITRADE(I,J) 00010810
C
C
  GOTO 150              00010820
C
C
C
C
C      DISPLAY MENU, PROBLEM STATUS, AND TRADEOFF INFORMATION 00010830
C
C
C
C
C
150 CONTINUE             00010840
  CALL NEWPAG            00010850
  CALL ANMODE            00010860
  CALL HOME              00010870
  WRITE(6,420) PNAME,SOLNO 00010880
  LARGE=999999
  IF(OBJ.EQ.'TOTAL DISTANCE') WRITE(6,406) TDIST,LARGE,LOAD(1), 00010890
  * LENGTH(1),LDDEV,LIMIT1,LOAD(2),LENGTH(2),LNDEV,LIMIT2,LOAD(3), 00011000
  * LENGTH(3)
  IF(OBJ.EQ.'LENGTH DEVIATION') WRITE(6,406) TDIST,LIMIT1,LOAD(1), 00011010
  * LENGTH(1),LDDEV,LIMIT2,LOAD(2),LENGTH(2),LNDEV,LARGE,LOAD(3), 00011020
  * LENGTH(3)
  IF(OBJ.EQ.'LOAD DEVIATION') WRITE(6,406) TDIST,LIMIT1,LOAD(1), 00011030
  * LENGTH(1),LDDEV,LARGE,LOAD(2),LENGTH(2),LNDEV,LIMIT2,LOAD(3), 00011040
  * LENGTH(3)
  IF(IROUTE.GE.4) WRITE(6,407) LOAD(4),LENGTH(4) 00011050
  IF(IROUTE.LT.4) WRITE(6,408) 00011060
  IF(IROUTE.GE.5) WRITE(6,409) LOAD(5),LENGTH(5) 00011070
  IF(IROUTE.LT.5) WRITE(6,410) 00011080
  IF(IROUTE.GE.6) WRITE(6,431) LOAD(6),LENGTH(6) 00011090
  IF(IROUTE.LT.6) WRITE(6,432) 00011100
  IF(IROUTE.GE.7) WRITE(6,435) LOAD(7),LENGTH(7) 00011110
  IF(IROUTE.LT.7) WRITE(6,436) 00011120
  IF(IROUTE.GE.8) WRITE(6,433) LOAD(8),LENGTH(8) 00011130
  IF(IROUTE.LT.8) WRITE(6,434) 00011140
  IF(IROUTE.GE.9) WRITE(6,411) ((I,LOAD(I),LENGTH(I)),I=9,IROUTE) 00011150
  WRITE(6,437) (I,I=1,MINO(6,NT)) 00011160
  WRITE(6,438) OBJ,(-ITRADE(1,I),I=1,MINO(6,NT)) 00011170
  WRITE(6,439) CNSTR1,(ITRADE(2,I),I=1,MINO(6,NT)) 00011180
  WRITE(6,439) CNSTR2,(ITRADE(3,I),I=1,MINO(6,NT)) 00011190
  WRITE(6,455) NTRADE,NT 00011200
455  FORMAT(1H ,T12,'ORIGINAL TRADEOFFS',I4,' REDUCED TRADEOFFS',I4) 00011210
  WRITE(6,412) 00011220
C
C
C
C      READ MENU OPTION 00011230
C
C
C
C      READ(6,*) MENU 00011240
  IF(MENU.EQ.8) THEN 00011250
    CALL FINITT(0,700) 00011260
    STOP                00011270
  ENDIF                00011280

```

```

C          00011370
C          00011380
C          00011390
C MENU OPTION 1: MINIMIZE TOTAL DISTANCE
C          00011400
C          00011410
C          00011420
C          00011430
IF(MENU.EQ.1) THEN          00011440
    OBJ='TOTAL DISTANCE'      00011450
    CNSTR1='LOAD DEVIATION'    00011460
    CNSTR2='LENGTH DEVIATION'
    IF(OLDOBJ.NE.1) THEN      00011470
        OLDOBJ=1              00011480
        GOTO 160               00011490
    ENDIF
    OLDOBJ=1                 00011500
    WRITE(6,440)               00011510
    READ(5,*) NUM             00011520
    IF(NUM.EQ.0) GOTO 160     00011530
    ELSE                      00011540
    LDDVLM=LDEV+MAXO(0,I TRADE(2,NUM)) 00011550
    LNDVLM=LDEV+MAXO(0,I TRADE(3,NUM)) 00011560
    LIMIT1=LDDVLM             00011570
    LIMIT2=LNDVLM             00011580
    LIMIT1=LDDVLM             00011590
    IF(I TRADE(6,NUM).EQ.0) CALL XCHNG2(I TRADE(4,NUM),I TRADE(5,NUM)) 00011600
    IF(I TRADE(6,NUM).NE.0) CALL XCHNG3(I TRADE(4,NUM),I TRADE(5,NUM),
*                                         I TRADE(6,NUM),I TRADE(7,NUM)) 00011620
    IMAXLN=-99                00011630
    IMAXLD=-99                00011640
    IMINLN=999999               00011650
    IMINLD=999999               00011660
    DO 151 K=1,IROUTE         00011670
        IF(LOKK(DEPOT(K)).NE.0) GOTO 151
        IF(LENGTH(K).GT.IMAXLN) IMAXLN=LENGTH(K)           00011680
        IF(LENGTH(K).LT.IMINLN.AND.LENGTH(K).GT.0) IMINLN=LENGTH(K) 00011690
        IF(LOAD(K).GT.IMAXLD) IMAXLD=LOAD(K)             00011700
        IF(LOAD(K).LT.IMINLD.AND.LOAD(K).GT.0) IMINLD=LOAD(K) 00011710
151   CONTINUE                  00011720
    MAXLN=IMAXLN              00011730
    MAXLD=IMAXLD              00011740
    MINLN=IMINLN              00011750
    MINLD=IMINLD              00011760
    GOTO 161                  00011770
160   WRITE(6,414) OBJ          00011780
    WRITE(6,415) CNSTR1        00011790
    READ(5,*) LDDVLM           00011800
    LIMIT1=LDDVLM             00011810
    WRITE(6,415) CNSTR2        00011820
    READ(5,*) LNDVLM           00011830
    LIMIT2=LNDVLM             00011840
    CALL NEWPAG                00011850
161   WRITE(6,404) OBJ,CNSTR1,LIMIT1,CNSTR2,LIMIT2 00011860
    DO 2000 IRUN=1,3           00011870
        IF(IRUN.EQ.1) CALL TWOOPT(O)
        TDIST=0
        DO 180 I=1,IROUTE       00011880
            TDIST=TDIST+LENGTH(IROUT)
180   DSTRXL=FLOAT(TDIST)*O.1*FLOAT(IRUN-1) 00011890
        CALL THROPT(DSTRXL)    00011900
        TDIST=0
        DO 181 I=1,IROUTE       00011910
            TDIST=TDIST+LENGTH(I)
181   IF(IRUN.EQ.1.OR.TDIST.LT.BESTDS) THEN 00011920
        BNTRAD=NTRADE
        DO 190 I=1,NTRADE       00011930
        DO 190 J=1,7             00011940
190   BTRADE(J,I)=TRADE(J,I) 00011950
        TDIST=0
        DO 200 I=1,IROUTE       00011960
            TDIST=TDIST+LENGTH(I)
            BESTLD(I)=LOAD(I)
            BESTLN(I)=LENGTH(I)
200   00011970
        00011980
        00011990
        00012000
        00012010
        00012020
        00012030
        00012040
        00012050
        00012060
        00012070

```

```

200      BDEPOT(I)=DEPOT(I)          00012080
        BESTDS=TDIST              00012090
        BMAXLD=MAXLD              00012100
        BMINLD=MINLD              00012110
        BMINLN=MINLN              00012120
        BMAXLN=MAXLN              00012130
        DO 201 I=1,NCITY+IROUTE    00012140
          BESTP(I)=PRED(I)         00012150
          BESTS(I)=SUCC(I)         00012160
          BESTTR(I)=TRUCK(I)       00012170
          BCUMLN(I)=CUMLN(I)       00012180
          BCUMLD(I)=CUMLD(I)       00012190
201      ENDIF                      00012200
2000     CONTINUE                   00012210
          TDIST=BESTDS           00012220
          MAXLD=BMAXLD            00012230
          MINLD=BMINLD            00012240
          MAXLN=BMAXLN            00012250
          MINLN=BMINLN            00012260
          DO 203 I=1,IROUTE        00012270
            LOAD(I)=BESTLD(I)       00012280
            LENGTH(I)=BESTLN(I)     00012290
203      DEPOT(I)=BDEPOT(I)         00012300
          DO 204 I=1,NCITY+IROUTE    00012310
            PRED(I)=BESTP(I)         00012320
            SUCC(I)=BESTS(I)         00012330
            TRUCK(I)=BESTTR(I)       00012340
            CUMLN(I)=BCUMLN(I)       00012350
204      CUMLD(I)=BCUMLD(I)         00012360
          NTRADE=BNTRAD           00012370
          DO 205 I=1,NTRADE         00012380
          DO 205 J=1,7              00012390
205      TRADE(J,I)=BTRADE(J,I)     00012400
          IF(NTRADE.GT.0) CALL VSORA(TRADE,7,7,NTRADE,1,WK,IER) 00012410
          NT=0                      00012420
          IF(NTRADE.EQ.0) GOTO 93    00012430
C          ELSE                     00012440
          PREV1=999999.              00012450
          PREV2=-999999.             00012460
          PREV3=-999999.             00012470
          DO 207 I=1,NTRADE         00012480
            IF(TRADE(1,I).NE.PREV1) THEN
              NT=NT+1                00012490
              ITRADE(1,NT)=TRADE(1,I) 00012500
              ITRADE(2,NT)=TRADE(2,I) 00012510
              ITRADE(3,NT)=TRADE(3,I) 00012520
              ITRADE(4,NT)=TRADE(4,I) 00012530
              ITRADE(5,NT)=TRADE(5,I) 00012540
              ITRADE(6,NT)=TRADE(6,I) 00012550
              ITRADE(7,NT)=TRADE(7,I) 00012560
              PREV1=TRADE(1,I)         00012570
              PREV2=TRADE(2,I)         00012580
              PREV3=TRADE(3,I)         00012590
              GOTO 207                00012600
            ENDIF                     00012610
            IF(TRADE(2,I).LE.PREV2.AND.TRADE(3,I).LE.PREV3) THEN
              ITRADE(2,NT)=TRADE(2,I) 00012620
              ITRADE(3,NT)=TRADE(3,I) 00012630
              ITRADE(4,NT)=TRADE(4,I) 00012640
              ITRADE(5,NT)=TRADE(5,I) 00012650
              ITRADE(6,NT)=TRADE(6,I) 00012660
              ITRADE(7,NT)=TRADE(7,I) 00012670
              PREV1=TRADE(1,I)         00012680
              PREV2=TRADE(2,I)         00012690
              PREV3=TRADE(3,I)         00012700
            ENDIF                     00012710
207      CONTINUE                   00012720
          CALL NONDOM(ITRADE,NT)     00012730
          GOTO 93                     00012740
        ENDIF                      00012750
C

```

```

C          00012790
C          00012800
C MENU OPTION 2: MINIMIZE ROUTE-LOAD DEVIATION 00012810
C          00012820
C          00012830
C
C IF(MENU.EQ.2) THEN 00012840
C   OBJ='LOAD DEVIATION'
C   CNSTR1='TOTAL DISTANCE'
C   CNSTR2='LENGTH DEVIATION'
C   IF(OLDOBJ.NE.2) THEN 00012850
C     OLDOBJ=2
C     GOTO 208
C   ENDIF
C   OLDOBJ=2 00012860
C   WRITE(6,440)
C   READ(5,*) NUM 00012870
C   IF(NUM.EQ.0) GOTO 208
C   ELSE 00012880
C     DLIMIT=TDIST+MAXO(0,ITRADE(2,NUM))
C     LNDVLM=LNDEV+MAXO(0,ITRADE(3,NUM))
C     LIMIT1=DLIMIT 00012890
C     LIMIT2=LNDVLM 00012900
C     IP1=ITRADE(4,NUM) 00012910
C     IP2=ITRADE(5,NUM) 00012920
C     IP3=ITRADE(6,NUM) 00012930
C     ITYPE=ITRADE(7,NUM) 00012940
C     IF(ITYPE.LE.4) CALL XCHNG3(IP1,IP2,IP3,ITYPE) 00012950
C     IF(ITYPE.EQ.5) CALL XCHNG2(IP1,IP2) 00012960
C     IF(ITYPE.EQ.6) CALL XCHNG2(IP2,IP3) 00012970
C     IF(ITYPE.EQ.7) CALL XCHNG2(IP1,IP3) 00012980
C     DO 2002 IRT=1,IROUTE 00012990
C       START=DEPOT(IRT)
C       IF(SUCC(START).GT.NCITY.OR.SUCC(START)).GT.NCITY 00013000
C         GOTO 2002 00013010
C       *
C       NEXT=SUCC(START) 00013020
C       NODE=NEXT 00013030
C       IF(SUCC(NODE).GT.NCITY) THEN 00013040
C         END=NODE 00013050
C         CALL TSP(START,END,PRED,SUCC,LNGTH,DIST,ALLOW) 00013060
C         LENGTH(IRT)=LNGTH 00013070
C         GOTO 2002 00013080
C       ENDIF 00013090
C       NEXT=SUCC(NODE) 00013100
C       GOT0 2001 00013110
C
C 2001      NODE=NEXT 00013120
C           IF(SUCC(NODE).GT.NCITY) THEN 00013130
C             END=NODE 00013140
C             CALL TSP(START,END,PRED,SUCC,LNGTH,DIST,ALLOW) 00013150
C             LENGTH(IRT)=LNGTH 00013160
C             GOTO 2002 00013170
C           ENDIF 00013180
C           NEXT=SUCC(NODE) 00013190
C           GOT0 2001 00013200
C
C 2002      CONTINUE 00013210
C           NEXT=NCITY+1 00013220
C           INSIDE=0 00013230
C
C 2003      NODE=NEXT 00013240
C           IF(NODE.EQ.NCITY+1.AND.INSIDE.EQ.1) GOTO 2004 00013250
C           INSIDE=1 00013260
C           IF(NODE.GT.NCITY) THEN 00013270
C             ITRK=TRUCK(NODE) 00013280
C             ILD=0 00013290
C             ILN=0 00013300
C           ENDIF 00013310
C           ILD=ILD+DEMAND(NODE) 00013320
C           IF(NODE.LE.NCITY) ILN=ILN+DIST(NODE,PRED(NODE)) + ALLOW 00013330
C           CUMLD(NODE)=ILD 00013340
C           CUMLN(NODE)=ILN 00013350
C           TRUCK(NODE)=ITRK 00013360
C           FEASLD(ITRK)=ILD 00013370
C           LOAD(ITRK)=ILD 00013380
C           NEXT=SUCC(NODE) 00013390
C           IF(NEXT.GT.NCITY) FEASLN(ITRK)=ILN+DIST(NODE,NEXT) 00013400
C           IF(NEXT.GT.NCITY) LENGTH(ITRK)=ILN+DIST(NODE,NEXT) 00013410
C           GOTO 2003 00013420
C
C 2004      TDIST=0 00013430
C           DO 2005 I=1,IROUTE 00013440
C           TDIST=TDIST+LENGTH(I) 00013450
C           IMAXLN=-99 00013460
C           IMAXLD=-99 00013470
C
C 2005      TDIST=TDIST+LENGTH(I) 00013480
C
C           IMAXLN=-99 00013490
C           IMAXLD=-99

```

```

IMINLN=999999          00013500
IMINLD=999999          00013510
DO 206 K=1,IROUTE      00013520
  IF(LOKK(DEPOT(K)).NE.0) GOTO 206
  IF(LENGTH(K).GT.IMAXLN) IMAXLN=LENGTH(K)          00013530
  IF(LENGTH(K).LT.IMINLN.AND.LENGTH(K).GT.0) IMINLN=LENGTH(K) 00013540
  IF(LOAD(K).GT.IMAXLD) IMAXLD=LOAD(K)            00013550
  IF(LOAD(K).LT.IMINLD.AND.LOAD(K).GT.0) IMINLD=LOAD(K) 00013560
206  CONTINUE           00013570
MAXLN=IMAXLN           00013580
MAXLD=IMAXLD           00013590
MINLN=IMINLN           00013600
MINLD=IMINLD           00013610
GOTO 209               00013620
208  WRITE(6,414) OBJ   00013630
     WRITE(6,415) CNSTR1
     READ(5,*) DLIMIT
     LIMIT1=DLIMIT
     WRITE(6,415) CNSTR2
     READ(5,*) LNDVLM
     LIMIT2=LNDVLM
     CALL NEWPAG
209  WRITE(6,404) OBJ,CNSTR1,LIMIT1,CNSTR2,LIMIT2 00013720
NT=0
DO 213 IRUN=1,3        00013730
  IF(IRUN.EQ.1) CALL LDDV2(0)
  LDDEV=MAXLD-MINLD
  LDRXLX=FLOAT(LDDEV)*.5*FLOAT(IRUN-1)
  CALL LDDV3(LDRXLX)
  LDDEV=MAXLD-MINLD
  IF(IRUN.EQ.1.OR.LDDEV.LT.BLDDEV) THEN
    DO 210 I=1,IROUTE
      BESTLD(I)=LOAD(I)
      BESTLN(I)=LENGTH(I)
210  BDEPOT(I)=DEPOT(I)
    DO 211 I=1,NCITY+IROUTE
      BESTP(I)=PRED(I)
      BESTS(I)=SUCC(I)
      BESTTR(I)=TRUCK(I)
      BCUMLN(I)=CUMLN(I)
211  BCUMLD(I)=CUMLD(I)
      BNTRAD=NTRADE
      DO 212 I=1,NTRADE
        DO 212 J=1,7
          BTRADE(J,I)=TRADE(J,I)
212  BMAXLD=MAXLD
      BMINLD=MINLD
      BMAXLN=MAXLN
      BMINLN=MINLN
      LDDEV=MAXLD-MINLD
      BLDDEV=LDDEV
      ENDIF
213  CONTINUE
      LDDEV=BLDDEV
      MAXLD=BMAXLD
      MINLD=BMINLD
      MAXLN=BMAXLN
      MINLN=BMINLN
      DO 214 I=1,IROUTE
        LOAD(I)=BESTLD(I)
        LENGTH(I)=BESTLN(I)
214  DEPOT(I)=BDEPOT(I)
      DO 215 I=1,NCITY+IROUTE
        PRED(I)=BESTP(I)
        SUCC(I)=BESTS(I)
        TRUCK(I)=BESTTR(I)
        CUMLN(I)=BCUMLN(I)
215  CUMLD(I)=BCUMLD(I)
      NTRADE=BNTRAD
      DO 216 I=1,NTRADE
        DO 216 J=1,7

```

```

216      TRADE(J,I)=BTRADE(J,I)          00014210
        IF(NTRADE.GT.0) CALL VSORA(TRADE,7,7,NTRADE,1,WK,IER) 00014220
        NT=0          00014230
        IF(NTRADE.EQ.0) GOTO 93          00014240
        PREV1=999999.                  00014250
        PREV2=-999999.                 00014260
        PREV3=-999999.                 00014270
        DO 218 I=1,NTRADE             00014280
          IF(TRADE(1,I).NE.PREV1) THEN 00014290
            NT=NT+1
            ITRADE(1,NT)=TRADE(1,I)    00014310
            ITRADE(2,NT)=TRADE(2,I)    00014320
            ITRADE(3,NT)=TRADE(3,I)    00014330
            ITRADE(4,NT)=TRADE(4,I)    00014340
            ITRADE(5,NT)=TRADE(5,I)    00014350
            ITRADE(6,NT)=TRADE(6,I)    00014360
            ITRADE(7,NT)=TRADE(7,I)    00014370
            PREV1=TRADE(1,I)           00014380
            PREV2=TRADE(2,I)           00014390
            PREV3=TRADE(3,I)           00014400
            GOTO 218                  00014410
          ENDIF
          IF(TRADE(2,I).LE.PREV2.AND.TRADE(3,I).LE.PREV3) THEN 00014420
            ITRADE(2,NT)=TRADE(2,I)    00014430
            ITRADE(3,NT)=TRADE(3,I)    00014440
            ITRADE(4,NT)=TRADE(4,I)    00014450
            ITRADE(5,NT)=TRADE(5,I)    00014460
            ITRADE(6,NT)=TRADE(6,I)    00014470
            ITRADE(7,NT)=TRADE(7,I)    00014480
            PREV1=TRADE(1,I)           00014490
            PREV2=TRADE(2,I)           00014500
            PREV3=TRADE(3,I)           00014510
          ENDIF
218     CONTINUE                      00014520
        CALL NONDOM(ITRADE,NT)       00014530
        GOTO 93                         00014540
      ENDIF
C
C
C
C      MENU 'OPTION 3: MINIMIZE ROUTE-LENGTH DEVIATION'      00014550
C
C
C      IF(MENU.EQ.3) THEN                                     00014560
        OBJ='LENGTH DEVIATION'                                00014570
        CNSTR1='TOTAL DISTANCE'                               00014580
        CNSTR2='LOAD DEVIATION'                               00014590
        IF(OLDOBJ.NE.3) THEN                                 00014600
          OLDOBJ=3                                         00014610
          GOTO 220                                         00014620
        ENDIF
        OLDOBJ=3                                         00014630
        WRITE(6,440)                                       00014640
        READ(5,*) NUM                                     00014650
        IF(NUM.EQ.0) GOTO 220                           00014660
      ELSE
        DLIMIT=TDIST+MAXO(0,ITRADE(2,NUM))               00014670
        LDDVLM=LDDEV+MAXO(0,ITRADE(3,NUM))               00014680
        LIMIT1=DLIMIT                                     00014690
        LIMIT2=LDDVLM                                    00014700
        IP1=ITRADE(4,NUM)                                00014710
        IP2=ITRADE(5,NUM)                                00014720
        IP3=ITRADE(6,NUM)                                00014730
        ITYPE=ITRADE(7,NUM)                                00014740
        IF(ITYPE.LE.4) CALL XCHNG3(IP1,IP2,IP3,ITYPE)   00014750
        IF(ITYPE.EQ.5) CALL XCHNG2(IP1,IP2)              00014760
        IF(ITYPE.EQ.6) CALL XCHNG2(IP2,IP3)              00014770
        IF(ITYPE.EQ.7) CALL XCHNG2(IP1,IP3)              00014780
        DO 3002 IRT=1,IRDUTE                            00014790
          START=DEPOT(IRT)                             00014800
          IF(SUCC(START).GT.NCITY.OR.SUCC(SUCC(START)).GT.NCITY) 00014810

```

```

*                                         GOTO 3002      00014920
3001   NEXT=SUCC(START)                  00014930
       NODE=NEXT
       IF(SUCC(NODE).GT.NCITY) THEN      00014940
         END=NODE
         CALL TSP(START,END,PRED,SUCC,LNGTH,DIST,ALLOW)
         LENGTH(IRT)=LNGTH
         GOTO 3002
       ENDIF
       NEXT=SUCC(NODE)
       GOTO 3001
3002   CONTINUE
       NEXT=NCITY+1
       INSIDE=0
3003   NODE=NEXT
       IF(NODE.EQ.NCITY+1.AND.INSIDE.EQ.1) GOTO 3004
       INSIDE=1
       IF(NODE.GT.NCITY) THEN
         ITRK=TRUCK(NODE)
         ILD=0
         ILN=0
       ENDIF
       ILD=ILD+DEMAND(NODE)
       IF(NODE.LE.NCITY) ILN=ILN+DIST(NODE,PRED(NODE)) + ALLOW
       CUMLD(NODE)=ILD
       CUMLN(NODE)=ILN
       TRUCK(NODE)=ITRK
       FEASLD(ITRK)=ILD
       LOAD(ITRK)=ILD
       NEXT=SUCC(NODE)
       IF(NEXT.GT.NCITY) FEASLN(ITRK)=ILN+DIST(NODE,NEXT)
       IF(NEXT.GT.NCITY) LENGTH(ITRK)=ILN+DIST(NODE,NEXT)
       GOTO 3003
3004   TDIST=0
       DO 3005 I=1,IROUTE
3005   TDIST=TDIST+LENGTH(I)
       IMAXLN=-99
       IMAXLD=-99
       IMINLN=999999
       IMINLD=999999
       DO 219 K=1,IROUTE
         IF(LOKK(DEPOT(K)).NE.0) GOTO 219
         IF(LENGTH(K).GT.IMAXLN) IMAXLN=LENGTH(K)
         IF(LENGTH(K).LT.IMINLN.AND.LENGTH(K).GT.0) IMINLN=LENGTH(K)
         IF(LOAD(K).GT.IMAXLD) IMAXLD=LOAD(K)
         IF(LOAD(K).LT.IMINLD.AND.LOAD(K).GT.0) IMINLD=LOAD(K)
219    CONTINUE
       MAXLN=IMAXLN
       MAXLD=IMAXLD
       MINLN=IMINLN
       MINLD=IMINLD
       GOTO 221
220    WRITE(6,414) OBJ
       WRITE(6,415) CNSTR1
       READ(5,*) DLIMIT
       LIMIT1=DLIMIT
       WRITE(6,415) CNSTR2
       READ(5,*) LDDVLM
       LIMIT2=LDDVLM
       CALL NEWPAG
221    WRITE(6,404) OBJ,CNSTR1,LIMIT1,CNSTR2,LIMIT2
       NT=0
       CALL LNDV2(0)
       CALL LNDV3(0)
       BNTRAD=NTRADE
       DO 225 I=1,NTRADE
       DO 225 J=1,7
225    BTRADE(J,I)=TRADE(J,I)
       DO 227 I=1,IROUTE
         BESTLD(I)=LOAD(I)
         BESTLN(I)=LENGTH(I)

```

```

227      BDEPOT(I)=DEPOT(I)          00015630
      DO 228 I=1,NCITY+IROUTE      00015640
         BESTP(I)=PRED(I)          00015650
         BESTS(I)=SUCC(I)          00015660
         BESTTR(I)=TRUCK(I)        00015670
         BCUMLN(I)=CUMLN(I)        00015680
228      BCUMLD(I)=CUMLD(I)        00015690
         LNDEV=MAXLN-MINLN        00015700
         BMAXLD=MAXLD            00015710
         BMINLD=MINLD            00015720
         BMAXLN=MAXLN            00015730
         BMINLN=MINLN            00015740
         BLNDEV=LNDEV            00015750
         LNRLX=FLOAT(LNDEV)*0.5    00015760
         CALL LNDV3(LNRLX)         00015770
         LNDEV=MAXLN-MINLN        00015780
         IF(LNDEV.GE.BLNDEV) THEN   00015790
            LNDEV=BLNDEV          00015800
            MAXLD=BMAXLD          00015810
            MINLD=BMINLD          00015820
            MAXLN=BMAXLN          00015830
            MINLN=BMINLN          00015840
            DO 229 I=1,IROUTE       00015850
               LOAD(I)=BESTLD(I)     00015860
               LENGTH(I)=BESTLN(I)    00015870
229      DEPOT(I)=BDEPOT(I)        00015880
         DO 230 I=1,NCITY+IROUTE      00015890
            PRED(I)=BESTP(I)        00015900
            SUCC(I)=BESTS(I)        00015910
            TRUCK(I)=BESTTR(I)      00015920
            CUMLN(I)=BCUMLN(I)      00015930
230      CUMLD(I)=BCUMLD(I)        00015940
         NTRADE=BNTRAD           00015950
         DO 231 I=1,NTRADE         00015960
         DO 231 J=1,7              00015970
            TRADE(J,I)=BTRADE(J,I)  00015980
231      ENDIF
         IF(NTRADE.GT.0) CALL VSORA(TRADE,7,7,NTRADE,1,WK,IER) 00015990
         NT=0
         IF(NTRADE.EQ.0) GOTO 93  00016000
         PREV1=999999.             00016010
         PREV2=-999999.            00016020
         PREV3=-999999.            00016030
         DO 233 I=1,NTRADE        00016040
            IF(TRADE(1,I).NE.PREV1) THEN 00016050
               NT=NT+1
               ITRADE(1,NT)=TRADE(1,I)  00016060
               ITRADE(2,NT)=TRADE(2,I)  00016070
               ITRADE(3,NT)=TRADE(3,I)  00016080
               ITRADE(4,NT)=TRADE(4,I)  00016090
               ITRADE(5,NT)=TRADE(5,I)  00016100
               ITRADE(6,NT)=TRADE(6,I)  00016110
               ITRADE(7,NT)=TRADE(7,I)  00016120
               PREV1=TRADE(1,I)         00016130
               PREV2=TRADE(2,I)         00016140
               PREV3=TRADE(3,I)         00016150
               GOTO 233
            ENDIF
            IF(TRADE(2,I).LE.PREV2.AND.TRADE(3,I).LE.PREV3) THEN 00016160
               ITRADE(2,NT)=TRADE(2,I)  00016170
               ITRADE(3,NT)=TRADE(3,I)  00016180
               ITRADE(4,NT)=TRADE(4,I)  00016190
               ITRADE(5,NT)=TRADE(5,I)  00016200
               ITRADE(6,NT)=TRADE(6,I)  00016210
               ITRADE(7,NT)=TRADE(7,I)  00016220
               PREV1=TRADE(1,I)         00016230
               PREV2=TRADE(2,I)         00016240
               PREV3=TRADE(3,I)         00016250
            ENDIF
233      CONTINUE
         CALL NONDUM(ITRADE,NT)    00016260

```

```

      GOTO 93                               00016340
      ENDIF                                00016350
C
C
C
C   MENU OPTION 4:  MANUAL ROUTE ADJUSTMENT 00016360
C
C
C
C   IF(MENU.EQ.4) THEN                   00016370
      WRITE(6,442)                         00016380
      READ(5,*) I                           00016390
      CALL ADJUST(I)                      00016400
      LNDEV=MAXLN-MINLN                  00016410
      GOTO 150                            00016420
      ENDIF                                00016430
C
C
C
C   MENU OPTION 5:  DISPLAY A PRIOR SOLUTION 00016440
C
C
C
C   IF(MENU.EQ.5) THEN                   00016450
      WRITE(6,441)                         00016460
      READ(5,*) I                           00016470
      CALL DISPLA(I,IROUTE,NCITY,PNAME,XCOORD,YCOORD,TIME)
      GOTO 150                            00016480
      ENDIF                                00016490
C
C
C
C   MENU OPTION 6:  BACKTRACK TO A PRIOR SOLUTION 00016500
C
C
C
C   IF(MENU.EQ.6) THEN                   00016510
      WRITE(6,441)                         00016520
      READ(5,*) NUMBER                    00016530
      CALL BKTRAK(NUMBER,I TRADE,NT,OBJ,OLDOBJ,LIMIT1,LIMIT2,CNSTR1,
      *                                     CNSTR2) 00016540
      SOLNO=NUMBER-1                     00016550
      GOTO 93                            00016560
      ENDIF                                00016570
C
C
C
C   MENU OPTION 7:  REMOVE A ROUTE FROM CALCULATIONS. 00016580
C
C
C
C   IF(MENU.EQ.7) THEN                   00016590
      WRITE(6,442)                         00016600
      READ(5,*) I                           00016610
      CALL LOKK (I,LOKK,NCITY,TRUCK,IROUTE) 00016620
      MAXLN=-999999                         00016630
      MAXLD=-999999                         00016640
      MINLN=999999                          00016650
      MINLD=999999                          00016660
      DO 250 I=1,IROUTE                  00016670
        IF(LOKK(DEPOT(I)).EQ.1) GOTO 250
        IF(LOAD(I).GT.MAXLD) MAXLD=LOAD(I)
        IF(LENGTH(I).GT.MAXLN) MAXLN=LENGTH(I)
        IF(LOAD(I).LT.MINLD.AND.LOAD(I).NE.0) MINLD=LOAD(I)
        IF(LENGTH(I).LT.MINLN.AND.LENGTH(I).NE.0) MINLN=LENGTH(I)
250    CONTINUE                            00016680
      LDDEV=MAXLD-MINLD                  00016690
      LNDEV=MAXLN-MINLN                  00016700
      DO 260 I=1,NT                      00016710
        IP1=ITRADE(4,I)                  00016720
        IP2=ITRADE(5,I)                  00016730
        IP3=ITRADE(6,I)                  00016740
        IF(LOKK(IP1).EQ.1.OR.LOKK(IP2).EQ.1.OR.LOKK(IP3).EQ.1) THEN
          ITRADE(1,I)=9999999             00016750
          ITRADE(2,I)=9999999             00016760
          ITRADE(3,I)=9999999             00016770
      ENDIF                                00016780

```

```

260      CONTINUE          00017050
        GOTO 150          00017060
      ENDIF              00017070
100 FORMAT(A)          00017080
101 FORMAT(2I3,I5,I5,I3) 00017090
102 FORMAT(3(I3,1X),I4) 00017100
103 FORMAT(2I3,I4)      00017110
255 FORMAT(1H ,T4,I3)   00017120
301 FORMAT(1H ,T11,I3,T20,I3,T33,I3,T42,I4,T50,I2,T57,I2,T63,I4,
*2I9,2I4)             00017130
400 FORMAT(1H ,/////' FINAL SOLUTION?//'(Y/N) ') 00017140
402 FORMAT(1H ,'OBJECTIVE TO MINIMIZE'
*/,T2,'1. LENGTH DEVIATION'/T2,'2. LOAD DEVIATION') 00017150
404 FORMAT(1H ,A,' PROBLEM BEING SOLVED SUBJECT TO FOLLOWING LIMITS:'//00017180
*T4,A,I5,/T4,A,I5)   00017190
405 FORMAT(1H //'' SOLUTION NUMBER',I2//3X,A/' MINIMIZATION PROBLEM'//00017200
*' ROUTE LOAD LENGTH') 00017210
406 FORMAT(1H ,T19,'MAIN MENU',T43,' STATUS LIMIT',T66,'ROUTES:
*/'//T62,'# LOAD DIST'
*T10,'1. MINIMIZE TOTAL DISTANCE    --',I6,3X,I4,T62,'1',I5,I6/ 00017220
*T10,'2. MINIMIZE LOAD DEVIATION    --',I6,3X,I4,T62,'2',I5,I6/ 00017230
*T10,'3. MINIMIZE LENGTH DEVIATION  --',I6,3X,I4,T62,'3',I5,I6) 00017240
407 FORMAT(1H ,T10,'4. MANUAL ROUTE IMPROVEMENT',T62,'4',I5,I6) 00017250
408 FORMAT(1H ,T10,'4. MANUAL ROUTE IMPROVEMENT') 00017260
409 FORMAT(1H ,T10,'5. DISPLAY PREVIOUS SOLUTION',T62,'5',I5,I6) 00017270
410 FORMAT(1H ,T10,'5. DISPLAY PREVIOUS SOLUTION') 00017280
411 FORMAT(1H ,T61,I2,I5,I6) 00017290
412 FORMAT(1H ,T10,'SELECT FROM MENU') 00017300
413 FORMAT(I1)          00017310
414 FORMAT(1H ,'SPECIFY NEW LIMITS FOR ',A,' PROBLEM') 00017320
415 FORMAT(1H ,A)          00017330
420 FORMAT(1H ,T20,'WORKLOAD-BALANCED VEHICLE ROUTING PROGRAM'
*,T20,A,/T30,'SOLUTION NUMBER',I3,/) 00017340
430 FORMAT(1H ,T10,'HARDCOPY WANTED? (Y/N)') 00017350
431 FORMAT(1H ,T10,'6. BACKTRACK TO PREV. SOL.',T62,'6',I5,I6) 00017360
432 FORMAT(1H ,T10,'6. BACKTRACK TO PREV. SOL.') 00017370
433 FORMAT(1H ,T10,'8. EXIT',T62,'8',I5,I6) 00017380
434 FORMAT(1H ,T10,'8. EXIT') 00017390
435 FORMAT(1H ,T10,'7. REMOVE ROUTE FROM CALC.',T62,'7',I5,I6) 00017400
436 FORMAT(1H ,T10,'7. REMOVE ROUTE FROM CALC.') 00017410
437 FORMAT(1H ,T31,'ESTIMATED TRADEOFFS:'//T40,6I5) 00017420
438 FORMAT(1H ,T12,A,1X,'IMPROVEMENT',T41,6I5) 00017430
439 FORMAT(1H ,T12,A,1X,'RELAXATION',T41,6I5) 00017440
440 FORMAT(1H ,,'WHICH TRADEOFF # IS ACCEPTABLE?') 00017450
441 FORMAT(1H ,,'SOLUTION NUMBER?') 00017460
442 FORMAT(1H ,,'ROUTE NUMBER?') 00017470
      END                00017480
C                         00017490
C                         00017500
C                         00017510
C                         00017520
C                         00017530
C                         00017540
C                         00017550
C                         00017560
C                         00017570
C                         00017580
C                         00017590
C*****00017600
C                         00017610
C                         00017620
C
      SUBROUTINE TWOOPT(DSTRXL) 00017630
C                         00017640
C                         00017650
C THIS SUBROUTINE IMPLEMENTS THE 2-OPT DISTANCE MINIMIZATION 00017660
C ARC EXCHANGE ALGORITHM. 00017670
C                         00017680
C*****00017690
CHARACTER*1 MODE          00017700
CHARACTER*44 PNAME,IPLACE 00017710
INTEGER EUCLID,CITY,XCOORD(O:120),YCOORD(O:120),DEMAND(O:120) 00017720
INTEGER HEAD(120),TAIL(120),PRED(120),SUCC(120),ROUTES,TWGT,WTLIM 00017730
INTEGER DIST,ALLOW,TDIST,DISTLM,TRUCK,IR(100),IFLAG(40),
* PERMI(40),PERMJ(40) 00017740
                                         00017750

```

```

DOUBLE PRECISION DSEED                                         00017760
INTEGER START,END,POINT1,POINT2,D,FEASLD(20),FEASLN(20)    00017770
INTEGER FTRUCK,FWD,BACK,TEMTRK(120),CUMLD(120),CUMLN(120) 00017780
INTEGER FOUND,TRCNT,TRK,TAG,D1,D2,D3,D4,D5,D6,D7,D8    00017790
INTEGER FSTART,FEND,FPRED(120),FSUCC(120)                 00017800
INTEGER PERMPR(120),PERMSU(120),PERMTR(120)                00017810
INTEGER DLIMIT,FEAS,DEPOT(20),DSTRSX                      00017820
DIMENSION DIST(0:120,0:120),SAVING(3,6000),SORT(6000),PERMSV(40) 00017830
DIMENSION ISORT(6000),JSORT(6000),LOAD(120),TRUCK(120)      00017840
DIMENSION LENGTH(120),WORK(6),LOKK(120),TRADE(7,500)        00017850
COMMON IROUTE,NCITY,IRUN,START,END,WTLIM,DISTLM,ALLOW,NPERM, 00017860
*ICOUNT,LDDVLM,LNDVLM,DLIMIT,MAXLD,MINLD,MAXLN,MINLN,D1,D2,D3,D4, 00017870
*D5,D6,DEPOT,PRED,SUCC,TRUCK,DEMAND,LENGTH,LOAD,CUMLD,CUMLN,TEMTRK, 00017880
*FEASLD,FEASLN,PERMPR,PERMSU,PERMTR,ISORT,JSORT,SORT,DIST,XCOORD, 00017890
*YCOORD,LOKK,TRADE,NTRADE,DSEED                           00017900
C
C
DO 100 I=1,NCITY+IROUTE                                     00017910
  FPRED(I)=PRED(I)                                         00017920
100  FSUCC(I)=SUCC(I)                                       00017930
   DO 101 I=1,IROUTE                                     00017940
     FEASLD(I)=LOAD(I)                                     00017950
101  FEASLN(I)=LENGTH(I)                                    00017960
   START=GGUBFS(DSEED)*NCITY+1                         00017970
   END=PRED(START)                                         00017980
   POINT1=PRED(START)                                     00017990
1  POINT1=SUCC(POINT1)                                     00018000
   MODE='F'                                              00018010
   IF(POINT1.EQ.PRED(PRED(END))) RETURN                  00018020
   IF(LOKK(POINT1).EQ.1) GOTO 1                          00018030
   POINT2=SUCC(POINT1)                                     00018040
2  POINT2=SUCC(POINT2)                                     00018050
   IF(POINT2.EQ.END) GOTO 1                            00018060
   IF(LOKK(POINT2).EQ.1) GOTO 2                          00018070
   IF(TRUCK(POINT1).NE.TRUCK(POINT2)) MODE='B'          00018080
C
C
C   DISTANCE REDUCTION TEST.
C
C
IS1=SUCC(POINT1)                                           00018100
IS2=SUCC(POINT2)                                           00018110
D1=DIST(POINT1,IS1)                                         00018120
D2=DIST(POINT2,IS2)                                         00018130
D3=DIST(POINT1,POINT2)                                     00018140
D4=DIST(IS1,IS2)                                           00018150
IF(D1+D2+DSTRSX.LE.D3+D4) GOTO 2                      00018160
IF(DSTRSX.NE.0.AND.TRUCK(POINT1).EQ.TRUCK(POINT2)) GOTO 2 00018170
C
C
C   FEASIBILITY TEST.
C
C
CALL FEAS2(POINT1,POINT2,FEAS,0,0,DSTRSX)               00018180
IF(FEAS.EQ.0) GOTO 2                                     00018190
C
C
C   EXCHANGE ARCS.
C
C
CALL XCHNG2(POINT1,POINT2)                                00018200
C
C

```

```

C
C
C      ROTATE
C
C
C
C      20 START=END          00018470
C           END=PRED(START) 00018480
C           POINT1=END      00018490
C           DSTRLX=0        00018500
C           GOTO 1          00018510
C
C
C      END                  00018520
C
C
C
C*****SUBROUTINE THROPT(DSTRLX)          00018530
C
C
C      THIS SUBROUTINE IS USED TO IMPLEMENT A 3-OPT DISTANCE MINIMIZATION 00018630
C      ARC-EXCHANGE ALGORITHM.          00018640
C
C
C*****CHARACTER*1 MODE12,MODE13,MODE23          00018650
C      CHARACTER*44 PNAME,IPLACE          00018660
C      INTEGER FMAXLD,FMINLD,FMAXLN,FMINLN          00018670
C      INTEGER EUCLID,CITY,XCOORD(0:120),YCOORD(0:120),DEMAND(0:120) 00018680
C      INTEGER HEAD(120),TAIL(120),PRED(120),SUCC(120),ROUTES,TWGT,WTLIM 00018690
C      INTEGER DIST,ALLOW,TDIST,DISTLM,TRUCK,IR(100),IFLAG(40),          00018700
C      * PERMI(40),PERMJ(40)          00018710
C      DOUBLE PRECISION DSEED          00018720
C      INTEGER START,END,POINT1,POINT2,D,FEASLD(20),FEASLN(20)          00018730
C      INTEGER FTRUCK,FWD,BACK,TEMTRK(120),CUMLD(120),CUMLN(120)          00018740
C      INTEGER FOUND,TRCNT,TRK,TAG,D1,D2,D3,D4,D5,D6,D7,D8,DSTRLX          00018750
C      INTEGER FSTART,FEND,P1,P2,P3,FPRED(120),FSUCC(120)          00018760
C      INTEGER PERMPR(120),PERMSU(120),PERMTR(120),DIFF          00018770
C      INTEGER DLIMIT,DEPOT(20),FEAS,POINT3,TRY(120,120)          00018780
C      DIMENSION DIST(0:120,0:120),SAVING(3,6000),SORT(6000),PERMSV(40) 00018790
C      DIMENSION ISORT(6000),JSQRT(6000),LOAD(120),TRUCK(120)          00018800
C      DIMENSION LENGTH(120),WORK(6),LOKK(120)          00018810
C      DIMENSION TRADE(7,500)          00018820
C      COMMON IROUTE,NCITY,IRUN,START,END,WTLIM,DISTLM,ALLOW,NPERM,          00018830
C      *ICOUNT,LDDVLM,LNDVLM,DLIMIT,MAXLD,MINLD,MAXLN,MINLN,D1,D2,D3,D4, 00018840
C      *D5,D6,DEPOT,PRED,SUCC,TRUCK,DEMAND,LENGTH,LOAD,CUMLD,CUMLN,TEMTRK, 00018850
C      *FEASLD,FEASLN,PERMPR,PERMSU,PERMTR,ISORT,JSORT,SORT,DIST,XCOORD, 00018860
C      *YCOORD,LOKK,TRADE,NTRADE,DSEED          00018870
C
C
C
C      DATA TRY/14400*0/          00018880
C      DO 98 I=1,NCITY+IROUTE          00018890
C           FPRED(I)=PRED(I)          00018900
C      98   FSUCC(I)=SUCC(I)          00018910
C           DO 99 I=1,IROUTE          00018920
C               FEASLD(I)=LOAD(I)          00018930
C      99   FEASLN(I)=LENGTH(I)          00018940
C           LNDEV=MAXLN-MINLN          00018950
C           LDDEV=MAXLD-MINLD          00018960
C           NTRADE=0          00018970
C           START=GGUBFS(DSEED)*NCITY+1          00018980
C           END=PRED(START)          00018990
C           POINT1=PRED(START)          00019000
C           1 POINT1=SUCC(POINT1)          00019010
C           IF(POINT1.EQ.PRED(PRED(END))) RETURN          00019020
C           IF(LOKK(POINT1).EQ.1) GOTO 1          00019030
C           IS1=SUCC(POINT1)          00019040

```

```

D1=DIST(POINT1,IS1)          00019180
POINT2=POINT1                00019190
MODE12='F'                   00019200
MODE13='F'                   00019210
2 POINT2=SUCC(POINT2)        00019220
IF(POINT2.EQ.PRED(END)) GOTO 1 00019230
IF(LOKK(POINT2).EQ.1) GOTO 2 00019240
IS2=SUCC(POINT2)             00019250
D2=DIST(POINT2,IS2)           00019260
MODE23='F'                   00019270
POINT3=POINT2                00019280
3 POINT3=SUCC(POINT3)         00019290
IF(TRUCK(POINT1).NE.TRUCK(POINT2)) MODE12='B' 00019300
IF(TRUCK(POINT1).NE.TRUCK(POINT3)) MODE13='B' 00019310
IF(TRUCK(POINT2).NE.TRUCK(POINT3)) MODE23='B' 00019320
IF(POINT3.EQ.END) GOTO 2    00019330
IF(LOKK(POINT3).EQ.1) GOTO 3 00019340
IF(DSTRXL.NE.0.AND.TRUCK(POINT1).EQ.TRUCK(POINT2).AND. 00019350
*   TRUCK(POINT1).EQ.TRUCK(POINT3)) GOTO 3
IF(DSTRXL.GT.0) THEN        00019360
  IF(POINT1.GT.NCITY) GOTO 3 00019370
  IF(POINT2.GT.NCITY) GOTO 3 00019380
  IF(POINT3.GT.NCITY) GOTO 3 00019390
ENDIF                      00019400
IS3=SUCC(POINT3)             00019420
D3=DIST(POINT3,IS3)           00019430
C                           00019440
C                           00019450
C                           00019460
C                           00019470
C                           00019480
C TYPE 1 EXCHANGE DISTANCE REDUCTION TEST 00019490
C                           00019500
NTYPE=1                     00019510
D4=DIST(POINT1,POINT2)        00019520
D5=DIST(IS1,POINT3)           00019530
D6=DIST(IS2,IS3)              00019540
DIFF=D1+D2+D3-D4-D5-D6       00019550
IF(D1+D2+D3+DSTRXL.LE.D4+D5+D6) GOTO 5 00019560
C                           00019570
C                           00019580
C TYPE 1 EXCHANGE FEASIBILITY TEST 00019590
C                           00019600
ASSIGN 4 TO IRTN             00019610
GOTO 100                     00019620
4 IF(FEAS.EQ.1) GOTO 14      00019630
C                           00019640
C                           00019650
C                           00019660
C TYPE 2 EXCHANGE DISTANCE REDUCTION TEST 00019670
C                           00019680
5 NTYPE=2                     00019690
D4=DIST(POINT1,IS2)           00019700
D5=DIST(POINT3,POINT2)        00019710
D6=DIST(IS1,IS3)              00019720
DIFF=D1+D2+D3-D4-D5-D6       00019730
IF(D1+D2+D3+DSTRXL.LE.D4+D5+D6) GOTO 7 00019740
C                           00019750
C                           00019760
C TYPE 2 EXCHANGE FEASIBILITY TEST 00019770
C                           00019780
ASSIGN 6 TO IRTN             00019790
GOTO 100                     00019800
6 IF(FEAS.EQ.1) GOTO 14      00019810
C                           00019820
C                           00019830
C                           00019840
C TYPE 3 EXCHANGE DISTANCE REDUCTION TEST 00019850
C                           00019860
7 NTYPE=3                     00019870
D4=DIST(POINT1,IS2)           00019880

```

```

D5=DIST(POINT3,IS1)          00019890
D6=DIST(POINT2,IS3)          00019900
DIFF=D1+D2+D3-D4-D5-D6      00019910
IF(D1+D2+D3+DSTRUX.LE.D4+D5+D6) GOTO 9  00019920
C                                     00019930
C                                     00019940
C                                     00019950
C                                     00019960
C                                     00019970
C                                     00019980
C                                     00019990
C                                     00020000
C                                     00020010
C                                     00020020
C                                     00020030
C                                     00020040
C                                     00020050
C                                     00020060
C                                     00020070
C                                     00020080
C                                     00020090
C                                     00020100
C                                     00020110
C                                     00020120
C                                     00020130
C                                     00020140
C                                     00020150
C                                     00020160
C                                     00020170
C                                     00020180
C                                     00020190
C                                     00020200
C                                     00020210
C                                     00020220
C                                     00020230
C                                     00020240
C                                     00020250
C                                     00020260
C                                     00020270
C                                     00020280
C                                     00020290
C                                     00020300
C                                     00020310
C                                     00020320
C                                     00020330
C                                     00020340
C                                     00020350
C                                     00020360
C                                     00020370
C                                     00020380
C                                     00020390
C                                     00020400
C                                     00020410
C                                     00020420
C                                     00020430
C                                     00020440
C                                     00020450
C                                     00020460
C                                     00020470
C                                     00020480
C                                     00020490
C                                     00020500
C                                     00020510
C                                     00020520
C                                     00020530
C                                     00020540
C                                     00020550
C                                     00020560
C                                     00020570
C                                     00020580
C                                     00020590

C                                     TYPE 3 EXCHANGE FEASIBILITY TEST
C                                     ASSIGN 8 TO IRTN
C                                     GOTO 100
C                                     IF(FEAS.EQ.1) GOTO 14
C                                     TYPE 4 EXCHANGE DISTANCE REDUCTION TEST
C                                     NTYPE=4
C                                     D4=DIST(POINT1,POINT3)
C                                     D5=DIST(IS2,IS1)
C                                     D6=DIST(POINT2,IS3)
C                                     DIFF=D1+D2+D3-D4-D5-D6
C                                     IF(D1+D2+D3+DSTRUX.LE.D4+D5+D6) GOTO 11
C                                     TYPE 4 EXCHANGE FEASIBILITY TEST
C                                     ASSIGN 10 TO IRTN
C                                     GOTO 100
C                                     IF(FEAS.EQ.1) GOTO 14
C                                     TYPE 5 EXCHANGE DISTANCE REDUCTION TEST
C                                     NTYPE=5
C                                     D4=DIST(POINT1,POINT2)
C                                     D5=DIST(IS1,IS2)
C                                     DIFF=D1+D2-D4-D5
C                                     IF(D1+D2+DSTRUX.LE.D4+D5) GOTO 12
C                                     TYPE 5 EXCHANGE FEASIBILITY TEST
C                                     IF(TRUCK(POINT1).EQ.TRUCK(POINT2).AND.DSTRUX.NE.0) GOTO 12
C                                     IF(TRY(POINT1,POINT2).EQ.0) THEN
C                                       CALL FEAS2(POINT1,POINT2,FEAS,NTYPE,DIFF,DSTRUX)
C                                       TRY(POINT1,POINT2)=1
C                                       IF(FEAS.EQ.1) GOTO 14
C                                     ENDIF
C                                     TYPE 6 EXCHANGE DISTANCE REDUCTION TEST
C                                     NTYPE=6
C                                     D4=DIST(POINT2,POINT3)
C                                     D5=DIST(IS2,IS3)
C                                     DIFF=D2+D3-D4-D5
C                                     IF(D2+D3+DSTRUX.LE.D4+D5) GOTO 13
C                                     TYPE 6 EXCHANGE FEASIBILITY TEST
C                                     IF(TRUCK(POINT2).EQ.TRUCK(POINT3).AND.DSTRUX.NE.0) GOTO 13
C                                     IF(TRY(POINT2,POINT3).EQ.0) THEN
C                                       CALL FEAS2(POINT2,POINT3,FEAS,NTYPE,DIFF,DSTRUX)
C                                       TRY(POINT2,POINT3)=1
C                                       IF(FEAS.EQ.1) GOTO 14
C                                     ENDIF

```

```

C          TYPE 7 EXCHANGE DISTANCE REDUCTION TEST          00020600
C          D4=DIST(POINT1,POINT3)                          00020610
C          D5=DIST(IS1,IS3)                                00020620
C          DIFF=D1+D3-D4-D5                            00020630
C          IF(D1+D3+DSTRXL.LE.D4+D5) GOTO 3            00020640
C
C          TYPE 7 EXCHANGE FEASIBILITY TEST             00020650
C
C          IF(TRUCK(POINT1).EQ.TRUCK(POINT3).AND.DSTRXL.NE.0) GOTO 3 00020660
C          IF(TRY(POINT1,POINT3).EQ.1) GOTO 3           00020670
C          IF(TRY(POINT1,POINT3).EQ.0) THEN              00020680
C              CALL FEAS2(POINT1,POINT3,FEAS,NTYPE,DIFF,DSTRXL) 00020690
C              TRY(POINT1,POINT3)=1                     00020700
C              IF(FEAS.EQ.0) GOTO 3                   00020710
C          ENDIF
C
C          PERFORM ARC EXCHANGE                      00020720
C
C          14 IF(NTYPE.LE.4) CALL XCHNG3(POINT1,POINT2,POINT3,NTYPE) 00020730
C              IF(NTYPE.EQ.5) CALL XCHNG2(POINT1,POINT2)        00020740
C              IF(NTYPE.EQ.6) CALL XCHNG2(POINT2,POINT3)        00020750
C              IF(NTYPE.EQ.7) CALL XCHNG2(POINT1,POINT3)        00020760
C
C          ROTATE                                     00020770
C
C          IRLX=DSTRXL                               00020780
C          DSTRXL=0                                  00020790
C          NTRADE=0                                 00020800
C          LDDEV=MAXLD-MINLD                         00020810
C          LNDEV=MAXLN-MINLN                         00020820
C          START=END                                00020830
C          IF(IRLX.GT.0) START=DEPOT(TRUCK(PRED(START))) 00020840
C          END=PRED(START)                           00020850
C          POINT1=END                                00020860
C          DO 15 I=1,NCITY+IRROUTE                  00020870
C          DO 15 J=1,NCITY+IRROUTE                  00020880
C          15 TRY(I,J)=0                            00020890
C          GOTO 1                                   00020900
C
C          100 CONTINUE                               00020910
C          999 FORMAT(1H , 'INSIDE FEAS3',4I5)       00020920
C          FEAS=0                                    00020930
C          P1=POINT1                                00020940
C          P2=POINT2                                00020950
C          P3=POINT3                                00020960
C
C          GOTO (110,120,130,140), NTYPE            00020970
C
C          TYPE 1 EXCHANGE                           00020980
C
C          IF ALL 3 POINTS ARE IN THE SAME ROUTE, THE EXCHANGE IS FEASIBLE. 00020990
C
C          110 IF(TRUCK(P1).EQ.TRUCK(P2).AND.TRUCK(P2).EQ.TRUCK(P3)) THEN 00021000
C              FEASLN(TRUCK(P1))=LENGTH(TRUCK(P1))+D4+D5+D6-D1-D2-D3 00021010
C              FEASLD(TRUCK(P1))=LOAD(TRUCK(P1))                    00021020
C              GOTO 150                                     00021030
C          ENDIF                                         00021040

```

```

C          00021310
C          00021320
C EACH POINT IN A DIFFERENT ROUTE:          00021330
C          00021340
C IF(TRUCK(P1).NE.TRUCK(P2).AND.TRUCK(P2).NE.TRUCK(P3).AND.TRUCK(P1)00021350
* .NE.TRUCK(P3)) THEN          00021360
C          00021370
C 1ST ROUTE:          00021380
IF(CUMLD(P1)+CUMLD(P2).GT.WTLIM) GOTO IRTN          00021390
IF(CUMLN(P1)+CUMLN(P2)+DIST(P1,P2).GT.DISTLM) GOTO IRTN          00021400
C 2ND ROUTE:          00021410
IF(LOAD(TRUCK(P1))-CUMLD(P1)+CUMLD(P3).GT.WTLIM) GOTO IRTN          00021420
L1=LENGTH(TRUCK(P1))-CUMLN(IS1)+ALLOW          00021430
IF(IS1.GT.NCITY) L1=0          00021440
IF(L1+CUMLN(P3)+DIST(IS1,P3).GT.DISTLM) GOTO IRTN          00021450
C 3RD ROUTE:          00021460
IF(LOAD(TRUCK(P2))-CUMLD(P2)+LOAD(TRUCK(P3))-CUMLD(P3).GT.WTLIM)00021470
* GOTO IRTN          00021480
L2=LENGTH(TRUCK(P2))-CUMLN(IS2)+ALLOW          00021490
IF(IS2.GT.NCITY) L2=0          00021500
L3=LENGTH(TRUCK(P3))-CUMLN(IS3)+ALLOW          00021510
IF(IS3.GT.NCITY) L3=0          00021520
IF(L2+L3+DIST(IS2,IS3).GT.DISTLM) GOTO IRTN          00021530
C          00021540
FEASLD(TRUCK(P1))=CUMLD(P1)+CUMLD(P2)          00021550
FEASLD(TRUCK(P2))=LOAD(TRUCK(P1))-CUMLD(P1)+CUMLD(P3)          00021560
FEASLD(TRUCK(P3))=LOAD(TRUCK(P2))-CUMLD(P2)+LOAD(TRUCK(P3))          00021570
* -CUMLD(P3)          00021580
FEASLN(TRUCK(P1))=CUMLN(P1)+CUMLN(P2)+DIST(P1,P2)          00021590
FEASLN(TRUCK(P2))=L1+CUMLN(P3)+DIST(IS1,P3)          00021600
FEASLN(TRUCK(P3))=L2+L3+DIST(IS2,IS3)          00021610
GOTO 150          00021620
ENDIF          00021630
C          00021640
C          00021650
C P1 IN ONE ROUTE; P2 & P3 IN OTHER ROUTE:          00021660
C          00021670
C IF(TRUCK(P2).EQ.TRUCK(P3).AND.TRUCK(P2).NE.TRUCK(P1)) THEN          00021680
C          00021690
C 1ST ROUTE:          00021700
IF(CUMLD(P1)+CUMLD(P2).GT.WTLIM) GOTO IRTN          00021710
IF(CUMLN(P1)+CUMLN(P2)+DIST(P1,P2).GT.DISTLM) GOTO IRTN          00021720
C 2ND ROUTE:          00021730
IF(LOAD(TRUCK(P1))-CUMLD(P1)+LOAD(TRUCK(P3))-CUMLD(P2).GT.WTLIM)00021740
* GOTO IRTN          00021750
L1=LENGTH(TRUCK(P1))-CUMLN(IS1)+ALLOW          00021760
IF(IS1.GT.NCITY) L1=0          00021770
L2=CUMLN(P3)-CUMLN(IS2)+ALLOW          00021780
L3=LENGTH(TRUCK(P3))-CUMLN(IS3)+ALLOW          00021790
IF(IS3.GT.NCITY) L3=0          00021800
IF(L1+L2+L3+DIST(IS1,P3).GT.DISTLM) GOTO IRTN          00021810
C          00021820
FEASLD(TRUCK(P1))=CUMLD(P1)+CUMLD(P2)          00021830
FEASLD(TRUCK(P2))=LOAD(TRUCK(P1))-CUMLD(P1)+LOAD(TRUCK(P3))          00021840
* -CUMLD(P2)          00021850
FEASLN(TRUCK(P1))=CUMLN(P1)+CUMLN(P2)+DIST(P1,P2)          00021860
FEASLN(TRUCK(P2))=L1+L2+L3+DIST(IS1,P3)+DIST(IS2,IS3)          00021870
GOTO 150          00021880
ENDIF          00021890
C          00021900
C          00021910
C P3 IN ONE ROUTE; P1 & P2 IN OTHER ROUTE:          00021920
C          00021930
C IF(TRUCK(P1).EQ.TRUCK(P2).AND.TRUCK(P1).NE.TRUCK(P3)) THEN          00021940
C          00021950
C 1ST ROUTE:          00021960
IF(CUMLD(P2)+CUMLD(P3).GT.WTLIM) GOTO IRTN          00021970
IF(CUMLN(P2)-DIST(P1,IS1)+CUMLN(P3)+DIST(P1,P2)+DIST(IS1,P3)00021980
* .GT.DISTLM) GOTO IRTN          00021990
C 2ND ROUTE:          00022000
IF(LOAD(TRUCK(P2))-CUMLD(P2)+LOAD(TRUCK(P3))-CUMLD(P3).GT.WTLIM)00022010

```

```

* GOTO IRTN
L1=LENGTH(TRUCK(P2))-CUMLN(IS2)+ALLOW
IF(IS2.GT.NCITY) L1=0
L2=LENGTH(TRUCK(P3))-CUMLN(IS3)+ALLOW
IF(IS3.GT.NCITY) L2=0
IF(L1+L2+DIST(IS2,IS3).GT.DISTLM) GOTO IRTN

C
FEASLD(TRUCK(P1))=CUMLD(P2)+CUMLD(P3)
FEASLD(TRUCK(P3))=LOAD(TRUCK(P2))-CUMLD(P2)+LOAD(TRUCK(P3))
* -CUMLD(P3)
FEASLN(TRUCK(P1))=CUMLN(P2)-DIST(P1,IS1)+CUMLN(P3)+DIST(P1,P2)
* +DIST(IS1,P3)
FEASLN(TRUCK(P3))=L1+L2+DIST(IS2,IS3)
GOTO 150
ENDIF

C
C
P2 IN ONE ROUTE; P1 & P3 IN OTHER ROUTE:
C
IF(TRUCK(P1).EQ.TRUCK(P3).AND.TRUCK(P2).NE.TRUCK(P1)) THEN
C
1ST ROUTE:
IF(LOAD(TRUCK(P2))+CUMLD(P1)-CUMLD(P3).GT.WTLIM) GOTO IRTN
L1=CUMLN(P1)-CUMLN(IS3)+ALLOW
L2=LENGTH(TRUCK(P2))-DIST(P2,IS2)
IF(L1+L2+DIST(P1,P2)+DIST(IS3,IS2).GT.DISTLM) GOTO IRTN
C
2ND ROUTE:
L3=LENGTH(TRUCK(P1))-CUMLN(IS1)+ALLOW
IF(IS1.GT.NCITY) L3=0
IF(L3+CUMLN(P3)+DIST(IS1,P3).GT.DISTLM) GOTO IRTN
C
FEASLD(TRUCK(P2))=LOAD(TRUCK(P2))+CUMLD(P1)-CUMLD(P3)
FEASLD(TRUCK(P1))=LOAD(TRUCK(P1))-CUMLD(P1)+CUMLD(P3)
FEASLN(TRUCK(P2))=L1+L2+DIST(P1,P2)+DIST(IS3,IS2)
FEASLN(TRUCK(P1))=L3+CUMLN(P3)+DIST(IS1,P3)
GOTO 150
ENDIF

C
C
C
C
TYPE 2 EXCHANGE
C
C
IF ALL 3 POINTS ARE IN THE SAME ROUTE, THE EXCHANGE IS FEASIBLE.

C
120 IF(TRUCK(P1).EQ.TRUCK(P2).AND.TRUCK(P2).EQ.TRUCK(P3)) THEN
C
FEASLN(TRUCK(P1))=LENGTH(TRUCK(P1))+D4+D5+D6-D1-D2-D3
FEASLD(TRUCK(P1))=LOAD(TRUCK(P1))
GOTO 150
ENDIF

C
C
EACH POINT IN A DIFFERENT ROUTE:
C
IF(TRUCK(P1).NE.TRUCK(P2).AND.TRUCK(P2).NE.TRUCK(P3).AND.
* TRUCK(P1).NE.TRUCK(P3)) THEN
C
1ST ROUTE:
IF(CUMLD(P1)+LOAD(TRUCK(P2))-CUMLD(P2).GT.WTLIM) GOTO IRTN
L1=LENGTH(TRUCK(P2))-CUMLN(IS2)+ALLOW
IF(IS2.GT.NCITY) L1=0
IF(L1+CUMLN(P1)+DIST(P1,IS2).GT.DISTLM) GOTO IRTN
C
2ND ROUTE:
IF(CUMLD(P2)+CUMLD(P3).GT.WTLIM) GOTO IRTN
IF(CUMLN(P2)+CUMLN(P3)+DIST(P2,P3).GT.DISTLM) GOTO IRTN
C
3RD ROUTE:
IF(LOAD(TRUCK(P1))-CUMLD(P1)+LOAD(TRUCK(P3))-CUMLD(P3).GT.WTLIM)
* GOTO IRTN
L2=LENGTH(TRUCK(P1))-CUMLN(IS1)+ALLOW

```

```

IF(IS1.GT.NCITY) L2=0                                00022730
L3=LENGTH(TRUCK(P3))-CUMLN(IS3)+ALLOW             00022740
IF(IS3.GT.NCITY) L3=0                                00022750
IF(L2+L3+DIST(IS1,IS3).GT.DISTLM) GOTO IRTN       00022760
C
FEASLD(TRUCK(P1))=CUMLD(P1)+LOAD(TRUCK(P2))-CUMLD(P2) 00022770
FEASLD(TRUCK(P2))=CUMLD(P2)+CUMLD(P3)              00022780
FEASLD(TRUCK(P3))=LOAD(TRUCK(P1))-CUMLD(P1)+LOAD(TRUCK(P3)) 00022790
*          -CUMLD(P3)                                00022800
FEASLN(TRUCK(P1))=L1+CUMLN(P1)+DIST(P1,IS2)        00022810
FEASLN(TRUCK(P2))=CUMLN(P2)+CUMLN(P3)+DIST(P2,P3)  00022820
FEASLN(TRUCK(P3))=L2+L3+DIST(IS1,IS3)              00022830
GOTO 150                                            00022840
ENDIF                                              00022850
C
C
P1 IN ONE ROUTE; P2 & P3 IN OTHER ROUTE:           00022860
C
IF(TRUCK(P2).EQ.TRUCK(P3).AND.TRUCK(P1).NE.TRUCK(P2)) THEN 00022870
C
1ST ROUTE:
  IF(CUMLD(P1)+CUMLD(P3).GT.WTLIM) GOTO IRTN      00022880
  IF(CUMLN(P1)+CUMLN(P3)-DIST(P2,IS2)+DIST(P2,P3)+DIST(P1,IS2) 00022890
*          .GT.DISTLM) GOTO IRTN                   00022900
C
2ND ROUTE:
  IF(LOAD(TRUCK(P1))-CUMLD(P1)+LOAD(TRUCK(P2))-CUMLD(P3).GT.WTLIM) 00022910
*          GOTO IRTN                               00022920
  L1=LENGTH(TRUCK(P1))-CUMLN(IS1)+ALLOW            00022930
  IF(IS1.GT.NCITY) L1=0                            00022940
  L2=LENGTH(TRUCK(P2))-CUMLN(IS3)+ALLOW            00022950
  IF(IS3.GT.NCITY) L2=0                            00022960
  IF(L1+L2+DIST(IS1,IS3).GT.DISTLM) GOTO IRTN     00022970
C
FEASLD(TRUCK(P1))=CUMLD(P1)+CUMLD(P3)              00022980
FEASLD(TRUCK(P2))=LOAD(TRUCK(P1))-CUMLD(P1)+LOAD(TRUCK(P2)) 00022990
*          -CUMLD(P3)                                00023000
FEASLN(TRUCK(P1))=CUMLN(P1)+CUMLN(P3)-DIST(P2,IS2)+DIST(P2,P3) 00023010
*          +DIST(P1,IS2)                            00023020
FEASLN(TRUCK(P2))=L1+L2+DIST(IS1,IS3)              00023030
GOTO 150                                            00023040
ENDIF                                              00023050
C
C
P3 IN ONE ROUTE; P1 & P2 IN OTHER ROUTE:           00023060
C
IF(TRUCK(P1).EQ.TRUCK(P2).AND.TRUCK(P3).NE.TRUCK(P1)) THEN 00023070
C
1ST ROUTE:
  L1=LENGTH(TRUCK(P2))-CUMLN(IS2)+ALLOW            00023080
  IF(IS2.GT.NCITY) L1=0                            00023090
  IF(L1+CUMLN(P1)+DIST(P1,IS2).GT.DISTLM) GOTO IRTN 00023100
C
2ND ROUTE:
  IF(LOAD(TRUCK(P3))+CUMLD(P2)-CUMLD(P1).GT.WTLIM) 00023110
  IF(LENGTH(TRUCK(P3))-DIST(P3,IS3)+CUMLN(P2)-CUMLN(IS1)+ALLOW 00023120
*          +DIST(P2,P3)+DIST(IS1,IS3).GT.DISTLM) GOTO IRTN 00023130
C
FEASLD(TRUCK(P1))=LOAD(TRUCK(P1))-CUMLD(P2)+CUMLD(P1) 00023140
FEASLD(TRUCK(P3))=LOAD(TRUCK(P3))+CUMLD(P2)-CUMLD(P1) 00023150
FEASLN(TRUCK(P1))=L1+CUMLN(P1)+DIST(P1,IS2)          00023160
FEASLN(TRUCK(P3))=LENGTH(TRUCK(P3))-DIST(P3,IS3)+CUMLN(P2) 00023170
*          -CUMLN(IS1)+DIST(P2,P3)+DIST(IS1,IS3)+ALLOW 00023180
GOTO 150                                            00023190
ENDIF                                              00023200
C
C
P2 IN ONE ROUTE; P1 & P3 IN OTHER ROUTE:           00023210
C
IF(TRUCK(P1).EQ.TRUCK(P3).AND.TRUCK(P1).NE.TRUCK(P2)) THEN 00023220
C
1ST ROUTE:
  IF(LOAD(TRUCK(P1))-CUMLD(P3)+LOAD(TRUCK(P2))-CUMLD(P2).GT.WTLIM) 00023230
*          00023240

```

```

*      GOTD IRTN                                00023440
L1=LENGTH(TRUCK(P1))-CUMLN(IS1)+ALLOW        00023450
IF(IS1.GT.NCITY) L1=0                          00023460
L2=LENGTH(TRUCK(P2))-CUMLN(IS2)+ALLOW        00023470
IF(IS2.GT.NCITY) L2=0                          00023480
IF(CUMLN(P1)-DIST(P3,IS3)+L1+L2+DIST(P1,IS2) 00023490
*      .GT.DISTLM) GOTO IRTN                  00023500
C      2ND ROUTE:                                00023510
    IF(CUMLD(P3)+CUMLD(P2).GT.WTLIM) GOTO IRTN 00023520
    IF(CUMLN(P3)+CUMLN(P2)+DIST(P2,P3).GT.DISTLM) GOTO IRTN 00023530
C      FEASLD(TRUCK(P1))=LOAD(TRUCK(P1))-CUMLD(P3)+LOAD(TRUCK(P2)) 00023550
*      -CUMLD(P2)                            00023560
FEASLD(TRUCK(P2))=CUMLD(P3)+CUMLD(P2)        00023570
FEASLN(TRUCK(P1))=CUMLN(P1)-DIST(P3,IS3)+L1+L2+DIST(IS1,IS3) 00023580
*      +DIST(P1,IS2)                          00023590
FEASLN(TRUCK(P2))=CUMLN(P3)+CUMLN(P2)+DIST(P2,P3)            00023600
GOTO 150                                      00023610
ENDIF                                         00023620
C                                         00023630
C                                         00023640
C                                         00023650
C                                         00023660
C      TYPE 3 EXCHANGE                         00023670
C                                         00023680
C                                         00023690
C      IF ALL 3 POINTS ARE IN THE SAME ROUTE, THE EXCHANGE IS FEASIBLE. 00023700
C                                         00023710
130 IF(TRUCK(P1).EQ.TRUCK(P2).AND.TRUCK(P1).EQ.TRUCK(P3)) THEN 00023720
C                                         00023730
    FEASLN(TRUCK(P1))=LENGTH(TRUCK(P1))+D4+D5+D6-D1-D2-D3 00023740
    FEASLD(TRUCK(P1))=LOAD(TRUCK(P1))                      00023750
    GOTO 150                                         00023760
    ENDIF                                         00023770
C                                         00023780
C                                         00023790
C      EACH POINT IN A DIFFERENT ROUTE:          00023800
C                                         00023810
    IF(TRUCK(P1).NE.TRUCK(P2).AND.TRUCK(P2).NE.TRUCK(P3).AND.TRUCK(P1) 00023820
*      .NE.TRUCK(P3)) THEN                      00023830
C                                         00023840
C      1ST ROUTE:                                00023850
    IF(CUMLD(P1)+LOAD(TRUCK(P2))-CUMLD(P2).GT.WTLIM) GOTO IRTN 00023860
    L1=LENGTH(TRUCK(P2))-CUMLN(IS2)+ALLOW        00023870
    IF(IS2.GT.NCITY) L1=0                          00023880
    IF(CUMLN(P1)+L1+DIST(P1,IS2).GT.DISTLM) GOTO IRTN 00023890
C      2ND ROUTE:                                00023900
    IF(LOAD(TRUCK(P1))-CUMLD(P1)+CUMLD(P3).GT.WTLIM) GOTO IRTN 00023910
    L2=LENGTH(TRUCK(P1))-CUMLN(IS1)+ALLOW        00023920
    IF(IS1.GT.NCITY) L2=0                          00023930
    IF(L2+CUMLN(P3)+DIST(IS1,P3).GT.DISTLM) GOTO IRTN 00023940
C      3RD ROUTE:                                00023950
    IF(CUMLD(P2)+LOAD(TRUCK(P3))-CUMLD(P3).GT.WTLIM) GOTO IRTN 00023960
    L3=LENGTH(TRUCK(P3))-CUMLN(IS3)+ALLOW        00023970
    IF(IS3.GT.NCITY) L3=0                          00023980
    IF(CUMLN(P2)+L3+DIST(P2,IS3).GT.DISTLM) GOTO IRTN 00023990
C                                         00024000
    FEASLD(TRUCK(P1))=CUMLD(P1)+LOAD(TRUCK(P2))-CUMLD(P2) 00024010
    FEASLD(TRUCK(P2))=LOAD(TRUCK(P1))-CUMLD(P1)+CUMLD(P3) 00024020
    FEASLD(TRUCK(P3))=CUMLD(P2)+LOAD(TRUCK(P3))-CUMLD(P3) 00024030
    FEASLN(TRUCK(P1))=CUMLN(P1)+L1+DIST(P1,IS2)          00024040
    FEASLN(TRUCK(P2))=L2+CUMLN(P3)+DIST(IS1,P3)          00024050
    FEASLN(TRUCK(P3))=CUMLN(P2)+L3+DIST(P2,IS3)          00024060
    GOTO 150                                         00024070
    ENDIF                                         00024080
C                                         00024090
C      P1 IN ONE ROUTE; P2 & P3 IN OTHER ROUTE: 00024100
C                                         00024110
    IF(TRUCK(P2).EQ.TRUCK(P3).AND.TRUCK(P1).NE.TRUCK(P2)) THEN 00024120
C                                         00024130
C                                         00024140

```

```

C   1ST ROUTE:
    IF(LOAD(TRUCK(P1))+CUMLD(P3)-CUMLD(P2).GT.WTLIM) GOTO IRTN      00024150
    L1=CUMLN(P3)-CUMLN(IS2)+ALLOW                                         00024160
    IF(L1+LENGTH(TRUCK(P1))-DIST(P1,IS1)+DIST(P1,IS2)+DIST(P3,IS1)      00024170
    * .GT.DISTLM) GOTO IRTN                                              00024180
C   2ND ROUTE:
    L2=LENGTH(TRUCK(P3))-CUMLN(IS3)+ALLOW                                         00024190
    IF(IS3.GT.NCITY) L2=0                                                 00024200
    IF(L2+CUMLN(P2)+DIST(P2,IS3).GT.DISTLM) GOTO IRTN                      00024210
C
    FEASLD(TRUCK(P1))=LOAD(TRUCK(P1))+CUMLD(P3)-CUMLD(P2)                  00024220
    FEASLD(TRUCK(P2))=LOAD(TRUCK(P2))-CUMLD(P3)+CUMLD(P2)                  00024230
    FEASLN(TRUCK(P1))=L1+LENGTH(TRUCK(P1))-DIST(P1,IS1)+DIST(P1,IS2)      00024240
    * +DIST(P3,IS1)                                                       00024250
    FEASLN(TRUCK(P2))=L2+CUMLN(P2)+DIST(P2,IS3)                           00024260
    GOTO 150                                                               00024270
    ENDIF                                                                    00024280
C
C
C   P3 IN ONE ROUTE; P1 & P2 IN OTHER ROUTE:
C
    IF(TRUCK(P1).EQ.TRUCK(P2).AND.TRUCK(P1).NE.TRUCK(P3)) THEN           00024290
C
    1ST ROUTE:
        L1=LENGTH(TRUCK(P2))-CUMLN(IS2)+ALLOW                               00024300
        IF(IS2.GT.NCITY) L1=0                                               00024310
        IF(L1+CUMLN(P1)+DIST(P1,IS2).GT.DISTLM) GOTO IRTN                  00024320
C
    2ND ROUTE:
        IF(LOAD(TRUCK(P3))+CUMLD(P2)-CUMLD(P1).GT.WTLIM) GOTO IRTN      00024330
        L2=LENGTH(TRUCK(P3))-DIST(P3,IS3)+ALLOW                               00024340
        IF(L2+CUMLN(P2)-CUMLN(IS1)+DIST(P2,IS3)+DIST(P3,IS1)+ALLOW       00024350
        * .GT.DISTLM) GOTO IRTN                                              00024360
C
    FEASLD(TRUCK(P2))=LOAD(TRUCK(P2))-CUMLD(P2)+CUMLD(P1)                00024370
    FEASLD(TRUCK(P3))=LOAD(TRUCK(P3))+CUMLD(P2)-CUMLD(P1)                00024380
    FEASLN(TRUCK(P2))=L1+CUMLN(P1)+DIST(P1,IS2)                           00024390
    FEASLN(TRUCK(P3))=L2+CUMLN(P2)-CUMLN(IS1)+DIST(P2,IS3)+ALLOW       00024400
    * +DIST(P3,IS1)                                                       00024410
    GOTO 150                                                               00024420
    ENDIF                                                                    00024430
C
C
C   P2 IN ONE ROUTE; P1 & P3 IN OTHER ROUTE:
C
    IF(TRUCK(P1).EQ.TRUCK(P3).AND.TRUCK(P1).NE.TRUCK(P2)) THEN           00024440
C
    1ST ROUTE:
        IF(LOAD(TRUCK(P2))+CUMLD(P1)-CUMLD(P3).GT.WTLIM) GOTO IRTN      00024450
        L1=LENGTH(TRUCK(P2))-DIST(P2,IS2)                                     00024460
        IF(L1+CUMLN(P1)-CUMLN(IS3)+ALLOW+DIST(P2,IS3)+DIST(P1,IS2).GT. 00024470
        * DISTLM) GOTO IRTN                                              00024480
C
    2ND ROUTE:
        L2=LENGTH(TRUCK(P1))-CUMLN(IS1)+ALLOW                               00024490
        IF(IS1.GT.NCITY) L2=0                                               00024500
        IF(L2+CUMLN(P3)+DIST(P3,IS1).GT.DISTLM) GOTO IRTN                  00024510
C
    FEASLD(TRUCK(P2))=LOAD(TRUCK(P2))+CUMLD(P1)-CUMLD(P3)                 00024520
    FEASLD(TRUCK(P1))=LOAD(TRUCK(P1))-CUMLD(P1)+CUMLD(P3)                 00024530
    FEASLN(TRUCK(P2))=L1+CUMLN(P1)-CUMLN(IS3)+DIST(P2,IS3)+DIST(P1,IS2) 00024540
    * +ALLOW)00024550
    FEASLN(TRUCK(P1))=L2+CUMLN(P3)+DIST(P3,IS1)                           00024560
    GOTO 150                                                               00024570
    ENDIF                                                                    00024580
C
C
C   TYPE 4 EXCHANGE
C
    IF ALL 3 POINTS ARE IN THE SAME ROUTE, THE EXCHANGE IS FEASIBLE. 00024590

```

```

C          00024860
C 140 IF(TRUCK(P1).EQ.TRUCK(P2).AND.TRUCK(P1).EQ.TRUCK(P3)) THEN 00024870
C          00024880
C          FEASLN(TRUCK(P1))=LENGTH(TRUCK(P1))+D4+D5+D6-D1-D2-D3 00024890
C          FEASLD(TRUCK(P1))=LOAD(TRUCK(P1)) 00024900
C          GOTO 150 00024910
C          ENDIF 00024920
C          00024930
C          EACH POINT IN A DIFFERENT ROUTE: 00024940
C          00024950
C          IF(TRUCK(P1).NE.TRUCK(P2).AND.TRUCK(P2).NE.TRUCK(P3).AND.TRUCK(P1) 00024970
C          * .NE.TRUCK(P3)) THEN 00024980
C          00024990
C          1ST ROUTE: 00025000
C          IF(CUMLD(P1)+CUMLD(P3).GT.WTLIM) GOTO IRTN 00025010
C          IF(CUMLN(P1)+CUMLN(P3)+DIST(P1,P3).GT.DISTLM) GOTO IRTN 00025020
C          2ND ROUTE: 00025030
C          IF(LOAD(TRUCK(P1))-CUMLD(P1)+LOAD(TRUCK(P2))-CUMLD(P2).GT.WTLIM) 00025040
C          * GOTO IRTN 00025050
C          L1=LENGTH(TRUCK(P1))-CUMLN(IS1)+ALLOW 00025060
C          IF(IS1.GT.NCITY) L1=0 00025070
C          L2=LENGTH(TRUCK(P2))-CUMLN(IS2)+ALLOW 00025080
C          IF(IS2.GT.NCITY) L2=0 00025090
C          IF(L1+L2+DIST(IS2,IS1).GT.DISTLM) GOTO IRTN 00025100
C          3RD ROUTE: 00025110
C          IF(CUMLD(P2)+LOAD(TRUCK(P3))-CUMLD(P3).GT.WTLIM) GOTO IRTN 00025120
C          L3=LENGTH(TRUCK(P3))-CUMLN(IS3)+ALLOW 00025130
C          IF(IS3.GT.NCITY) L3=0 00025140
C          IF(CUMLN(P2)+L3+DIST(P2,IS3).GT.DISTLM) GOTO IRTN 00025150
C          00025160
C          FEASLD(TRUCK(P1))=CUMLD(P1)+CUMLD(P3) 00025170
C          FEASLD(TRUCK(P2))=LOAD(TRUCK(P1))-CUMLD(P1)+LOAD(TRUCK(P2)) 00025180
C          * -CUMLD(P2) 00025190
C          FEASLD(TRUCK(P3))=CUMLD(P2)+LOAD(TRUCK(P3))-CUMLD(P3) 00025200
C          FEASLN(TRUCK(P1))=CUMLN(P1)+CUMLN(P3)+DIST(P1,P3) 00025210
C          FEASLN(TRUCK(P2))=L1+L2+DIST(IS2,IS1) 00025220
C          FEASLN(TRUCK(P3))=CUMLN(P2)+L3+DIST(P2,IS3) 00025230
C          GOTO 150 00025240
C          ENDIF 00025250
C          00025260
C          P1 IN ONE ROUTE; P2 & P3 IN OTHER ROUTE: 00025270
C          00025280
C          IF(TRUCK(P2).EQ.TRUCK(P3).AND.TRUCK(P1).NE.TRUCK(P2)) THEN 00025300
C          00025310
C          1ST ROUTE: 00025320
C          IF(LOAD(TRUCK(P1))+CUMLD(P3)-CUMLD(P2).GT.WTLIM) GOTO IRTN 00025330
C          L1=LENGTH(TRUCK(P1))-CUMLN(IS1)+ALLOW 00025340
C          IF(IS1.GT.NCITY) L1=0 00025350
C          IF(CUMLN(P1)+L1+CUMLN(P3)-CUMLN(IS2)+ALLOW+DIST(P1,P3)+ 00025360
C          * DIST(IS2,IS1).GT.DISTLM) GOTO IRTN 00025370
C          2ND ROUTE: 00025380
C          L2=LENGTH(TRUCK(P3))-CUMLN(IS3)+ALLOW 00025390
C          IF(IS3.GT.NCITY) L2=0 00025400
C          IF(L2+CUMLN(P2)+DIST(P2,IS3).GT.DISTLM) GOTO IRTN 00025410
C          00025420
C          FEASLD(TRUCK(P1))=LOAD(TRUCK(P1))+CUMLD(P3)-CUMLD(P2) 00025430
C          FEASLD(TRUCK(P2))=LOAD(TRUCK(P2))-CUMLD(P3)+CUMLD(P2) 00025440
C          FEASLN(TRUCK(P1))=CUMLN(P1)+L1+CUMLN(P3)-CUMLN(IS2)+DIST(P1,P3) 00025450
C          * +DIST(IS2,IS1)+ALLOW 00025460
C          FEASLN(TRUCK(P2))=L2+CUMLN(P2)+DIST(P2,IS3) 00025470
C          GOTO 150 00025480
C          ENDIF 00025490
C          00025500
C          P3 IN ONE ROUTE; P1 & P2 IN OTHER ROUTE: 00025510
C          00025520
C          IF(TRUCK(P1).EQ.TRUCK(P2).AND.TRUCK(P3).NE.TRUCK(P2)) THEN 00025540
C          00025550
C          1ST ROUTE: 00025560

```

```

IF(CUMLD(P1)+CUMLD(P3).GT.WTLIM) GOTO IRTN
IF(CUMLN(P1)+CUMLN(P3)+DIST(P1,P3).GT.DISTLM) GOTO IRTN
C 2ND ROUTE:
*   IF(LOAD(TRUCK(P2))-CUMLD(P1)+LOAD(TRUCK(P3))-CUMLD(P3).GT.WTLIM) 00025600
*     GOTO IRTN
L1=LENGTH(TRUCK(P3))-CUMLN(IS3)+ALLOW
IF(IS3.GT.NCITY) L1=0
L2=LENGTH(TRUCK(P2))-CUMLN(IS2)+ALLOW
IF(IS2.GT.NCITY) L2=0
L3=CUMLN(P2)-CUMLN(IS1)+ALLOW
IF(L1+L2+L3+DIST(IS2,IS1)+DIST(P2,IS3).GT.DISTLM) GOTO IRTN
C
FEASLD(TRUCK(P1))=CUMLD(P1)+CUMLD(P3)
FEASLD(TRUCK(P3))=LOAD(TRUCK(P2))-CUMLD(P1)+LOAD(TRUCK(P3))
*           -CUMLD(P3)
FEASLN(TRUCK(P1))=CUMLN(P1)+CUMLN(P3)+DIST(P1,P3)
FEASLN(TRUCK(P3))=L1+L2+L3+DIST(IS2,IS1)+DIST(P2,IS3)
GOTO 150
ENDIF
C
C P2 IN ONE ROUTE; P1 & P3 IN OTHER ROUTE:
C
IF(TRUCK(P1).EQ.TRUCK(P3).AND.TRUCK(P1).NE.TRUCK(P2)) THEN
C
1ST ROUTE:
IF(CUMLD(P1)+CUMLD(P2).GT.WTLIM) GOTO IRTN
L1=CUMLN(P1)-CUMLN(IS3)+ALLOW
IF(L1+CUMLN(P3)+CUMLN(P2)+DIST(P1,P3)+DIST(P2,IS3).GT.DISTLM) 00025850
*     GOTO IRTN
C
2ND ROUTE:
IF(LOAD(TRUCK(P1))-CUMLD(P1)+LOAD(TRUCK(P2))-CUMLD(P2).GT.WTLIM) 00025880
*     GOTO IRTN
L2=LENGTH(TRUCK(P1))-CUMLN(IS1)+ALLOW
IF(IS1.GT.NCITY) L2=0
L3=LENGTH(TRUCK(P2))-CUMLN(IS2)+ALLOW
IF(IS2.GT.NCITY) L3=0
IF(L2+L3+DIST(IS2,IS1).GT.DISTLM) GOTO IRTN
C
FEASLD(TRUCK(P1))=CUMLD(P1)+CUMLD(P2)
FEASLD(TRUCK(P2))=LOAD(TRUCK(P1))-CUMLD(P1)+LOAD(TRUCK(P2))
*           -CUMLD(P2)
FEASLN(TRUCK(P1))=L1+CUMLN(P3)+CUMLN(P2)+DIST(P1,P3)+DIST(P2,IS3)
FEASLN(TRUCK(P2))=L2+L3+DIST(IS2,IS1)
GOTO 150
ENDIF
C
C
C
C
C
ROUTE LOAD DEVIATION AND ROUTE LENGTH DEVIATION TESTS:
C
C
150 DO 151 I=1,IROUTE
IF(I.NE.TRUCK(P1).AND.I.NE.TRUCK(P2).AND.I.NE.TRUCK(P3)) THEN
  FEASLD(I)=LOAD(I)
  FEASLN(I)=LENGTH(I)
ENDIF
151 CONTINUE
FMAXLD=-99
FMINLD=999999
DO 155 I=1,IROUTE
  IF(LOKK(DEPOT(I)).NE.O) GOTO 155
  IF(FEASLD(I).GT.FMAXLD) FMAXLD=FEASLD(I)
  IF(FEASLD(I).LT.FMINLD.AND.FEASLD(I).NE.O) FMINLD=FEASLD(I)
155 CONTINUE
C
FMAXLN=-99
FMINLN=999999

```

```

DO 160 I=1,IROUTE          00026280
  IF(LOKK(DEPOT(I)).NE.0) GOTO 160      00026290
  IF(FEASLN(I).GT.FMAXLN) FMAXLN=FEASLN(I) 00026300
  IF(FEASLN(I).LT.FMINLN.AND.FEASLN(I).NE.0.) FMINLN=FEASLN(I) 00026310
160 CONTINUE                00026320
C                               00026330
C                               00026340
C                               00026350
C TRADEOFF ANALYSIS          00026360
C                               00026370
C                               00026380
C IF(DSTRXL.GT.0) GOTO 161      00026390
C ELSE                         00026400
C IF(FMAXLD-FMINLD.GT.LDDVLM.OR.FMAXLN-FMINLN.GT.LNDVLM) THEN 00026410
  NTRADE=NTRADE+1              00026420
  IF(NTRADE.GT.500) NTRADE=500      00026430
  TRADE(1,NTRADE)=-DIFF        00026440
  TRADE(2,NTRADE)=FMAXLD-FMINLD-LDDEV 00026450
  TRADE(3,NTRADE)=FMAXLN-FMINLN-LNDEV 00026460
  TRADE(4,NTRADE)=P1            00026470
  TRADE(5,NTRADE)=P2            00026480
  TRADE(6,NTRADE)=P3            00026490
  TRADE(7,NTRADE)=NTYPE         00026500
ENDIF                         00026510
161 IF(FMAXLD-FMINLD.GT.LDDVLM) GOTO IRTN 00026520
  IF(FMAXLN-FMINLN.GT.LNDVLM) GOTO IRTN 00026530
C                               00026540
C                               00026550
C EXCHANGE IS FEASIBLE        00026560
C                               00026570
C FEAS=1                      00026580
  MAXLD=FMAXLD                00026590
  MINLD=FMINLD                00026600
  MAXLN=FMAXLN                00026610
  MINLN=FMINLN                00026620
  GOTO IRTN                   00026630
END                           00026640
C                               00026650
C                               00026660
C                               00026670
C                               00026680
C*****00026690
C SUBROUTINE XCHNG2(P1,P2)      00026710
C                               00026720
C                               00026730
C THIS SUBROUTINE PERFORMS ARC EXCHANGES FOR A 2-OPT ALGORITHM. 00026740
C                               00026750
C*****00026760
C                               00026770
C                               00026780
C CHARACTER*1 MODE             00026790
CHARACTER*44 PNAME, IPLACE    00026800
INTEGER P1,P2,DEPOT1,DEPOT2,DEPOT3,DEPOT4,STACK(100),HEAD1,TAIL2 00026810
INTEGER EUCLID,CITY,XCOORD(0:120),YCOORD(0:120),DEMAND(0:120) 00026820
INTEGER HEAD(120),TAIL(120),PRED(120),SUCC(120),ROUTES,TWGT,WTLIM 00026830
INTEGER DIST,ALLOW,TDIST,DISTLM,TRUCK,IR(100),IFLAG(40),        00026840
* PERMI(40),PERMJ(40)          00026850
DOUBLE PRECISION DSEED       00026860
INTEGER START,END,POINT1,POINT2,D,FEASLD(20),FEASLN(20)        00026870
INTEGER FTRUCK,FWD,BACK,TEMTRK(120),CUMLD(120),CUMLN(120) 00026880
INTEGER FOUND,TRCNT,TRK,TAG,D1,D2,D3,D4,D5,D6,D7,D8        00026890
INTEGER FSTART,FEND,FPRED(120),FSUCC(120)                    00026900
INTEGER PERMPR(120),PERMSU(120),PERMTR(120)                  00026910
INTEGER DLIMIT,DEPOT(20),GAP1,GAP2,TALE(20),FIRST           00026920
DIMENSION DIST(0:120,0:120),SAVING(3,6000),SORT(6000),PERMSV(40) 00026930
DIMENSION ISORT(6000),JSORT(6000),LOAD(120),TRUCK(120)        00026940
DIMENSION LENGTH(120),WORK(6),LOKK(120),TRADE(7,500)        00026950
COMMON IROUTE,NCITY,IRUN,START,END,WTLIM,DISTLM,ALLOW,NPERM,   00026960
*ICOUNT,LDDVLM,LNDVLM,DLIMIT,MAXLD,MINLD,MAXLN,MINLN,D1,D2,D3,D4, 00026970
*D5,D6,DEPOT,PRED,SUCC,TRUCK,DEMAND,LENGTH,LOAD,CUMLD,CUMLN,TEMTRK,00026980

```

```

*FEASLD, FEASLN, PERMPR, PERMSU, PERMTR, ISORT, JSORT, SORT, DIST, XCOORD, 00026990
*YCOORD, LOKK, TRADE, NTRADE, DSEED 00027000
C                                         00027010
C                                         00027020
C                                         00027030
C                                         00027040
C                                         00027050
C                                         00027060
C     NULL EXCHANGE:                   00027070
C                                         00027080
C                                         00027090
C                                         00027100
C                                         00027110
C                                         00027120
C                                         00027130
C                                         00027140
C                                         00027150
C     REMOVE THE ROUTES INVOLVED IN THE EXCHANGE FROM THE NETWORK AND 00027160
C     FORM A SEPARATE NETWORK WITHIN WHICH THE 2-ARC EXCHANGE WILL TAKE 00027170
C     PLACE.                           00027180
C                                         00027190
C                                         00027200
C                                         00027210
C                                         00027220
C                                         00027230
C                                         00027240
C                                         00027250
C                                         00027260
C                                         00027270
C                                         00027280
C                                         00027290
C                                         00027300
C                                         00027310
C                                         00027320
C                                         00027330
C                                         00027340
C                                         00027350
C                                         00027360
C                                         00027370
C                                         00027380
C                                         00027390
C                                         00027400
C                                         00027410
C                                         00027420
C                                         00027430
C                                         00027440
C                                         00027450
C                                         00027460
C                                         00027470
C                                         00027480
C                                         00027490
C                                         00027500
C                                         00027510
C                                         00027520
C                                         00027530
C                                         00027540
C                                         00027550
C                                         00027560
C                                         00027570
C                                         00027580
C                                         00027590
C                                         00027600
C                                         00027610
C                                         00027620
C                                         00027630
C                                         00027640
C                                         00027650
C                                         00027660
C                                         00027670
C                                         00027680
C                                         00027690
C
C     PERFORM THE EXCHANGE.

```

```

C
C
IS1=SUCC(P1)          00027700
IS2=SUCC(P2)          00027710
LAST=P1               00027720
NEXT=P2               00027730
SUCC(P1)=P2           00027740
7 NODE=NEXT           00027750
IP=PRED(NODE)         00027760
PRED(NODE)=LAST       00027770
IF(NODE.EQ.IS2) GOTO 8 00027780
SUCC(NODE)=IP         00027790
IF(NODE.EQ.IS1) SUCC(NODE)=IS2 00027800
LAST=NODE             00027810
NEXT=SUCC(NODE)       00027820
GOTO 7               00027830
C
C
C
C   CALCULATE ROUTE LENGTHS AND LOADS. 00027840
C
C
C
8 PRED(IS2)=LAST      00027850
NEXT=FIRST             00027860
INSIDE=0               00027870
9 NODE=NEXT           00027880
IF(NODE.EQ.FIRST.AND.INSIDE.EQ.1) GOTO 10 00027890
INSIDE=1               00027900
IF(NODE.GT.NCITY) THEN 00027910
  ITRK=TRUCK(NODE)
  LOAD(ITRK)=0
  LENGTH(ITRK)=0
  CUMLD(NODE)=0
  CUMLN(NODE)=0
ENDIF
TRUCK(NODE)=ITRK
LOAD(ITRK)=LOAD(ITRK)+DEMAND(NODE)
LENGTH(ITRK)=LENGTH(ITRK)+DIST(NODE,SUCC(NODE))
IF(SUCC(NODE).LE.NCITY) LENGTH(ITRK)=LENGTH(ITRK)+ALLOW
IF(NODE.LE.NCITY) THEN
  CUMLD(NODE)=CUMLD(PRED(NODE))+DEMAND(NODE)
  CUMLN(NODE)=CUMLN(PRED(NODE))+DIST(NODE,PRED(NODE))+ALLOW
ENDIF
NEXT=SUCC(NODE)
GOTO 9               00028000
C
C
C
C   RECONNECT ROUTES INVOLVED IN EXCHANGE BACK INTO ORIGINAL NETWORK. 00028010
C
C
C
10 IPRED=PRED(FIRST)  00028020
SUCC(GAP1)=FIRST       00028030
PRED(FIRST)=GAP1       00028040
SUCC(IPRED)=GAP2       00028050
PRED(GAP2)=IPRED       00028060
RETURN                00028070
END                   00028080
C
*****
C***** SUBROUTINE XCHNG3(P1,P2,P3,TYPE) 00028290
C
C
C   THIS SUBROUTINE PERFORMS ARC EXCHANGES FOR A 3-OPT ALGORITHM. 00028300
C
C
CHARACTER*44 PNAME,IPLACE          00028310
INTEGER EUCLID,CITY,XCOORD(0:120),YCOORD(0:120),DEMAND(0:120) 00028320
INTEGER HEAD(120),TAIL(120),PRED(120),SUCC(120),ROUTES,TWGT,WTLIM 00028330
INTEGER DIST,ALLOW,TDIST,DISTLM,TRUCK,IR(100),IFLAG(40),        00028340
                                         00028350
                                         00028360
                                         00028370
                                         00028380
                                         00028390
                                         00028400

```

```

* PERMI(40),PERMU(40)                                         00028410
DOUBLE PRECISION DSEED                                         00028420
INTEGER START,END,POINT1,POINT2,D,FEASLD(20),FEASLN(20)        00028430
INTEGER FTRUCK,FWD,BACK,TEMTRK(120),CUMLD(120),CUMLN(120)    00028440
INTEGER FOUND,TRCNT,TRK,TAG,D1,D2,D3,D4,D5,D6,D7,D8,P1,P2,P3 00028450
INTEGER FSTART,FEND,FPRED(120),FSUCC(120),GAP1,GAP2           00028460
INTEGER PERMPR(120),PERMSU(120),PERMTR(120),LOKK(120)         00028470
INTEGER DLIMIT,TYPE,PLINK(3),DPLINK(3),DEPOT(20),FIRST,TALE(120) 00028480
DIMENSION DIST(O:120,O:120),SAVING(3,6000),SORT(6000),PERMSV(40) 00028490
DIMENSION TRADE(7,500)                                         00028500
DIMENSION ISORT(6000),JSORT(6000),LOAD(120),TRUCK(120)        00028510
DIMENSION LENGTH(120),WORK(6),LINK1(3),LINK2(3)                00028520
COMMON IROUTE,NCITY,IRUN,START,END,WTLIM,DISTLM,ALLOW,NPERM,      00028530
*ICOUNT,LDDVLM,LNDVLM,DLIMIT,MAXLD,MINLD,MAXLN,MINLN,D1,D2,D3,D4, 00028540
*D5,D6,DEPOT,PRED,SUCC,TRUCK,DEMAND,LENGTH,LOAD,CUMLD,CUMLN,TEMTRK, 00028550
*FEASLD,FEASLN,PERMPR,PERMSU,PERMTR,ISORT,JSORT,SORT,DIST,XCOORD, 00028560
*YCOORD,LOKK,TRADE,NTRADE,DSEED                             00028570
C                                                               00028580
C                                                               00028590
C                                                               00028600
C                                                               00028610
C NULL EXCHANGE:                                              00028620
C                                                               00028630
C IF(P1.EQ.PRED(P2).AND.P2.EQ.PRED(P3).AND.TYPE.EQ.1) RETURN 00028640
C IF(TYPE.EQ.3.AND.P1.GT.NCITY.AND.P2.GT.NCITY.AND.P3.GT.NCITY) 00028650
*   RETURN                                                 00028660
C                                                               00028670
C                                                               00028680
C                                                               00028690
C                                                               00028700
C REMOVE THE ROUTES INVOLVED IN THE EXCHANGE FROM THE NETWORK AND 00028710
C FORM A SEPARATE NETWORK WITHIN WHICH THE 3-OPT EXCHANGE WILL TAKE 00028720
C PLACE.                                                 00028730
C                                                               00028740
C                                                               00028750
FIRST=DEPOT(TRUCK(P1))                                         00028760
IS1=SUCC(P1)                                               00028770
IS2=SUCC(P2)                                               00028780
IS3=SUCC(P3)                                               00028790
FWD=0                                                       00028800
BACK=1                                                       00028810
DO 1 I=NCITY+1,NCITY+IROUTE                                 00028820
1 TAIL(DEPOT(TRUCK(PRED(I))))=PRED(I)                      00028830
NEXT=NCITY+1                                               00028840
NUM=0                                                       00028850
INSIDE=0                                                   00028860
3 NODE=NEXT                                              00028870
IF(NODE.EQ.NCITY+1.AND.INSIDE.EQ.1) GOTO 4               00028880
INSIDE=1                                                 00028890
IF(TRUCK(NODE).EQ.TRUCK(P1).OR.TRUCK(NODE).EQ.TRUCK(P2).OR. 00028900
*   TRUCK(NODE).EQ.TRUCK(P3)) THEN                         00028910
NEXT=SUCC(TAIL(NODE))                                     00028920
GOTO 3                                                   00028930
ENDIF                                                       00028940
NUM=NUM+1                                                 00028950
HEAD(NUM)=NODE                                           00028960
TALE(NUM)=TAIL(NODE)                                     00028970
NEXT=SUCC(TAIL(NODE))                                     00028980
GOTO 3                                                   00028990
4 IF(TRUCK(P1).EQ.TRUCK(P2).AND.TRUCK(P1).EQ.TRUCK(P3)) THEN 00029000
   SUCC(TAIL(DEPOT(TRUCK(P1))))=DEPOT(TRUCK(P1))          00029010
   PRED(DEPOT(TRUCK(P1)))=TAIL(DEPOT(TRUCK(P1)))          00029020
   GOTO 5                                                 00029030
ENDIF                                                       00029040
IF(TRUCK(P1).NE.TRUCK(P2).AND.TRUCK(P2).NE.TRUCK(P3).AND. 00029050
*   TRUCK(P1).NE.TRUCK(P3)) THEN                         00029060
   SUCC(TAIL(DEPOT(TRUCK(P1))))=DEPOT(TRUCK(P2))          00029070
   PRED(DEPOT(TRUCK(P2)))=TAIL(DEPOT(TRUCK(P1)))          00029080
   SUCC(TAIL(DEPOT(TRUCK(P2))))=DEPOT(TRUCK(P3))          00029090
   PRED(DEPOT(TRUCK(P3)))=TAIL(DEPOT(TRUCK(P2)))          00029100
   SUCC(TAIL(DEPOT(TRUCK(P3))))=DEPOT(TRUCK(P1))          00029110

```



```

C
C
C
C      TYPE 2 EXCHANGE
C
C
20 LAST=P1
    NEXT=IS2
    SUCC(P1)=IS2
    MODE=FWD
21 NODE=NEXT
    IP=PRED(NODE)
    PRED(NODE)=LAST
    IF(NODE.EQ.IS3) GOTO 50
    IF(MODE.EQ.BACK) SUCC(NODE)=IP
    IF(NODE.EQ.P3) THEN
        SUCC(NODE)=P2
        MODE=BACK
    ENDIF
    IF(NODE.EQ.IS1) SUCC(NODE)=IS3
    LAST=NODE
    NEXT=SUCC(NODE)
    GOTO 21
C
C
C
C      TYPE 3 EXCHANGE
C
C
30 LAST=P1
    NEXT=IS2
    SUCC(P1)=IS2
31 NODE=NEXT
    PRED(NODE)=LAST
    IF(NODE.EQ.IS3) GOTO 50
    IF(NODE.EQ.P3) SUCC(NODE)=IS1
    IF(NODE.EQ.P2) SUCC(NODE)=IS3
    LAST=NODE
    NEXT=SUCC(NODE)
    GOTO 31
C
C
C
C
C      TYPE 4 EXCHANGE
C
C
40 LAST=P1
    NEXT=P3
    SUCC(P1)=P3
    MODE=BACK
41 NODE=NEXT
    IP=PRED(NODE)
    PRED(NODE)=LAST
    IF(NODE.EQ.IS3) GOTO 50
    IF(MODE.EQ.BACK) SUCC(NODE)=IP
    IF(NODE.EQ.IS2) THEN
        SUCC(NODE)=IS1
        MODE=FWD
    ENDIF
    IF(NODE.EQ.P2) SUCC(NODE)=IS3
    LAST=NODE
    NEXT=SUCC(NODE)
    GOTO 41
C
C
C
C
C      CALCULATION OF ROUTE LENGTHS AND LOADS.

```

00029830
00029840
00029850
00029860
00029870
00029880
00029890
00029900
00029910
00029920
00029930
00029940
00029950
00029960
00029970
00029980
00029990
00030000
00030010
00030020
00030030
00030040
00030050
00030060
00030070
00030080
00030090
00030100
00030110
00030120
00030130
00030140
00030150
00030160
00030170
00030180
00030190
00030200
00030210
00030220
00030230
00030240
00030250
00030260
00030270
00030280
00030290
00030300
00030310
00030320
00030330
00030340
00030350
00030360
00030370
00030380
00030390
00030400
00030410
00030420
00030430
00030440
00030450
00030460
00030470
00030480
00030490
00030500
00030510
00030520
00030530

```

C          00030540
C          00030550
C          00030560
C          00030570
C          00030580
C          00030590
C          00030600
C          00030610
C          00030620
C          00030630
C          00030640
C          00030650
C          00030660
C          00030670
C          00030680
C          00030690
C          00030700
C          00030710
C          00030720
C          00030730
C          00030740
C          00030750
C          00030760
C          00030770
C          00030780
C          00030790
C          00030800
C          00030810
C          00030820
C          00030830
C          00030840
C          00030850
C          00030860
C          00030870
C          00030880
C          00030890
C          00030900
C          00030910
C          00030920
C          00030930
C          00030940
C          00030950
C          00030960
C          00030970
C          00030980
C          00030990
C          00031000
C          00031010
C          00031020
C          00031030
C          00031040
C          00031050
C          00031060
C          00031070
C          00031080
C          00031090
C          00031100
C          ****00031110
C          00031120
C          00031130
C          00031140
C          00031150
C          00031160
C          00031170
C          ****00031180
C          CHARACTER*44 PNAME
C          DOUBLE PRECISION DSEED
C          INTEGER P1,P2,P3,FIRST,TAIL(120),TALE(20),HEAD(20),GAP1,GAP2,
C          *      TYPE,FWD,BACK,START,END,WTLIM,DISTLM,ALLOW,DLIMIT,
C          *      D1,D2,D3,D4,D5,D6,DEPOT(20),PRED(120),SUCC(120),
C          *      TRUCK(120),DEMAND(0:120),LENGTH(120),LOAD(120),
C          00031190
C          00031200
C          00031210
C          00031220
C          00031230
C          00031240

```

```

*      CUMLD(120),CUMLN(120),TEMTRK(120),FEASLD(20),FEASLN(20), 00031250
*      PERMPR(120),PERMSU(120),PERMTR(120),ISDRT(6000), 00031260
*      JSORT(6000),DIST(0:120,0:120),XCOORD(0:120), 00031270
*      YCOORD(0:120),LOKK(120),PDUM(120),SDUM(120) 00031280
DIMENSION SORT(6000),TRADE(7,500) 00031290
COMMON IROUTE,NCITY,IRUN,START,END,WTLIM,DISTLM,ALLOW,NPERM, 00031300
*I COUNT,LDDVLM,LNDVLM,DLIMIT,MAXLD,MINLD,MAXLN,MINLN,D1,D2,D3,D4, 00031310
*D5,D6,DEPOT,PDUM,SDUM,TRUCK,DEMAND,LENGTH,LOAD,CUMLD,CUMLN,TEMTRK, 00031320
*FEASLD,FEASLN,PERMPR,PERMSU,PERMTR,ISORT,JSORT,SORT,DIST,XCOORD, 00031330
*YCOORD,LOKK,TRADE,NTRADE,DSEED 00031340
C 00031350
C 00031360
C 00031370
C NULL EXCHANGE: 00031380
C 00031390
IF(P1.EQ.PRED(P2).AND.P2.EQ.PRED(P3).AND.TYPE.EQ.1) RETURN 00031400
IF(TYPE.EQ.3.AND.P1.GT.NCITY.AND.P2.GT.NCITY.AND.P3.GT.NCITY) 00031410
* RETURN 00031420
C 00031430
C 00031440
C 00031450
C 00031460
C REMOVE THE ROUTES INVOLVED IN THE EXCHANGE FROM THE NETWORK AND 00031470
C FORM A SEPARATE NETWORK WITHIN WHICH THE 3-OPT EXCHANGE WILL TAKE 00031480
C PLACE. 00031490
C 00031500
C 00031510
IS1=SUCC(P1) 00031520
IS2=SUCC(P2) 00031530
IS3=SUCC(P3) 00031540
FWD=0 00031550
BACK=1 00031560
DO 1 I=NCITY+1,NCITY+IRROUTE 00031570
1 TAIL(DEPOT(TRUCK(PRED(I))))=PRED(I) 00031580
NEXT=NCITY+1 00031590
NUM=0 00031600
INSIDE=0 00031610
3 NODE=NEXT 00031620
IF(NODE.EQ.NCITY+1.AND.INSIDE.EQ.1) GOTO 4 00031630
INSIDE=1 00031640
IF(TRUCK(NODE).EQ.TRUCK(P1).OR.TRUCK(NODE).EQ.TRUCK(P2).OR. 00031650
* TRUCK(NODE).EQ.TRUCK(P3)) THEN 00031660
NEXT=SUCC(TAIL(NODE)) 00031670
GOTO 3 00031680
ENDIF 00031690
NUM=NUM+1 00031700
HEAD(NUM)=NODE 00031710
TALE(NUM)=TAIL(NODE) 00031720
NEXT=SUCC(TAIL(NODE)) 00031730
GOTO 3 00031740
4 IF(TRUCK(P1).EQ.TRUCK(P2).AND.TRUCK(P1).EQ.TRUCK(P3)) THEN 00031750
SUCC(TAIL(DEPOT(TRUCK(P1))))=DEPOT(TRUCK(P1)) 00031760
PRED(DEPOT(TRUCK(P1)))=TAIL(DEPOT(TRUCK(P1))) 00031770
GOTO 5 00031780
ENDIF 00031790
IF(TRUCK(P1).NE.TRUCK(P2).AND.TRUCK(P2).NE.TRUCK(P3).AND. 00031800
* TRUCK(P1).NE.TRUCK(P3)) THEN 00031810
SUCC(TAIL(DEPOT(TRUCK(P1))))=DEPOT(TRUCK(P2)) 00031820
PRED(DEPOT(TRUCK(P2)))=TAIL(DEPOT(TRUCK(P1))) 00031830
SUCC(TAIL(DEPOT(TRUCK(P2))))=DEPOT(TRUCK(P3)) 00031840
PRED(DEPOT(TRUCK(P3)))=TAIL(DEPOT(TRUCK(P2))) 00031850
SUCC(TAIL(DEPOT(TRUCK(P3))))=DEPOT(TRUCK(P1)) 00031860
PRED(DEPOT(TRUCK(P1)))=TAIL(DEPOT(TRUCK(P3))) 00031870
GOTO 5 00031880
ENDIF 00031890
IF(TRUCK(P1).EQ.TRUCK(P2)) THEN 00031900
SUCC(TAIL(DEPOT(TRUCK(P1))))=DEPOT(TRUCK(P3)) 00031910
PRED(DEPOT(TRUCK(P3)))=TAIL(DEPOT(TRUCK(P1))) 00031920
SUCC(TAIL(DEPOT(TRUCK(P3))))=DEPOT(TRUCK(P1)) 00031930
PRED(DEPOT(TRUCK(P1)))=TAIL(DEPOT(TRUCK(P3))) 00031940
GOTO 5 00031950

```

```

ENDIF
IF(TRUCK(P2).EQ.TRUCK(P3)) THEN
  SUCC(TAIL(DEPOT(TRUCK(P1))))=DEPOT(TRUCK(P2))
  PRED(DEPOT(TRUCK(P2)))=TAIL(DEPOT(TRUCK(P1)))
  SUCC(TAIL(DEPOT(TRUCK(P2))))=DEPOT(TRUCK(P1))
  PRED(DEPOT(TRUCK(P1)))=TAIL(DEPOT(TRUCK(P2)))
  GOTO 5
ENDIF
IF(TRUCK(P1).EQ.TRUCK(P3)) THEN
  SUCC(TAIL(DEPOT(TRUCK(P1))))=DEPOT(TRUCK(P2))
  PRED(DEPOT(TRUCK(P2)))=TAIL(DEPOT(TRUCK(P1)))
  SUCC(TAIL(DEPOT(TRUCK(P2))))=DEPOT(TRUCK(P1))
  PRED(DEPOT(TRUCK(P1)))=TAIL(DEPOT(TRUCK(P2)))
ENDIF
C
C
      5 IF(NUM.EQ.1) THEN
        SUCC(TALE(1))=HEAD(1)
        PRED(HEAD(1))=TALE(1)
        GAP1=TALE(1)
        GAP2=HEAD(1)
      ENDIF
      IF(NUM.GT.1) THEN
        DO 6 I=1,NUM-1
        SUCC(TALE(I))=HEAD(I+1)
      6  PRED(HEAD(I+1))=TALE(I)
        SUCC(TALE(NUM))=HEAD(1)
        PRED(HEAD(1))=TALE(NUM)
        GAP1=TALE(1)
        GAP2=SUCC(TALE(1))
      ENDIF
C
C
C
C      PERFORM THE EXCHANGE
C
IS1=SUCC(P1)
IS2=SUCC(P2)
IS3=SUCC(P3)
GOTD (10,20,30,40),TYPE
C
C
C
C      TYPE 1 EXCHANGE
C
C
      10 LAST=P1
        NEXT=P2
        SUCC(P1)=P2
      11 NODE=NEXT
        IP=PRED(NODE)
        PRED(NODE)=LAST
        IF(NODE.EQ.IS3) GOTO 50
        SUCC(NODE)=IP
        IF(NODE.EQ.IS1) SUCC(NODE)=P3
        IF(NODE.EQ.IS2) SUCC(NODE)=IS3
        LAST=NODE
        NEXT=SUCC(NODE)
        GOTO 11
C
C
C
C      TYPE 2 EXCHANGE
C
C
      20 LAST=P1
        NEXT=IS2
        SUCC(P1)=IS2

```

```

        MODE=FWD          00032670
21 NODE=NEXT          00032680
        IP=PRED(NODE)    00032690
        PRED(NODE)=LAST   00032700
        IF(NODE.EQ.IS3) GOTO 50   00032710
        IF(MODE.EQ.BACK) SUCC(NODE)=IP   00032720
        IF(NODE.EQ.P3) THEN   00032730
            SUCC(NODE)=P2   00032740
            MODE=BACK       00032750
        ENDIF             00032760
        IF(NODE.EQ.IS1) SUCC(NODE)=IS3   00032770
        LAST=NODE         00032780
        NEXT=SUCC(NODE)   00032790
        GOTO 21           00032800

C
C
C
C
C      TYPE 3 EXCHANGE
C
C
C
C
30 LAST=P1           00032810
    NEXT=IS2          00032820
    SUCC(P1)=IS2      00032830
31 NODE=NEXT         00032840
    PRED(NODE)=LAST   00032850
    IF(NODE.EQ.IS3) GOTO 50   00032860
    IF(NODE.EQ.P3) SUCC(NODE)=IS1   00032870
    IF(NODE.EQ.P2) SUCC(NODE)=IS3   00032880
    LAST=NODE         00032890
    NEXT=SUCC(NODE)   00032900
    GOTO 31           00032910

C
C
C
C
C      TYPE 4 EXCHANGE
C
C
C
C
40 LAST=P1           00032920
    NEXT=P3            00032930
    SUCC(P1)=P3        00032940
    MODE=BACK          00032950
41 NODE=NEXT         00032960
    IP=PRED(NODE)     00032970
    PRED(NODE)=LAST   00032980
    IF(NODE.EQ.IS3) GOTO 50   00032990
    IF(MODE.EQ.BACK) SUCC(NODE)=IP   00033000
    IF(NODE.EQ.IS2) THEN   00033010
        SUCC(NODE)=IS1   00033020
        MODE=FWD         00033030
    ENDIF             00033040
    IF(NODE.EQ.P2) SUCC(NODE)=IS3   00033050
    LAST=NODE         00033060
    NEXT=SUCC(NODE)   00033070
    GOTO 41           00033080

C
C
C
C
C
C
C
C
50 CONTINUE          00033090
        PRED(IS3)=LAST   00033100
C
C
C
C
C
C
C
C

```

```

C RECONNECT THE ROUTES INVOLVED IN THE EXCHANGE BACK INTO THE      00033380
C ORIGINAL NETWORK.                                              00033390
C                                                               00033400
C                                                               00033410
C
C IPRED=PRED(DEPOT(TRUCK(P1)))                                     00033420
C SUCC(GAP1)=DEPOT(TRUCK(P1))                                       00033430
C PRED(DEPOT(TRUCK(P1)))=GAP1                                         00033440
C SUCC(IPRED)=GAP2                                                 00033450
C PRED(GAP2)=IPRED                                               00033460
C RETURN                                                       00033470
C END                                                               00033480
C                                                               00033490
C                                                               00033500
C*****                                                               00033510
C                                                               00033520
C SUBROUTINE SAVNGS                                              00033530
C                                                               00033540
C                                                               00033550
C THIS SUBROUTINE IS USED TO IMPLEMENT THE CONCURRENT VERSION      00033560
C OF CLARKE AND WRIGHT'S SAVINGS ALGORITHM FOR SOLVING THE      00033570
C VEHICLE ROUTING PROBLEM.                                         00033580
C                                                               00033590
C*****                                                               00033600
CHARACTER*44 PNAME,IPLACE                                         00033610
INTEGER EUCLID,CITY,XCOORD(0:120),YCOORD(0:120),DEMAND(0:120)    00033620
INTEGER HEAD(120),TAIL(120),PRED(120),SUCC(120),ROUTES,TWGT,WTLIM 00033630
INTEGER DIST,ALLOW,TDIST,DISTLM,TRUCK,IR(100),IFLAG(40),          00033640
* PERMI(40),PERMJ(40)                                            00033650
DOUBLE PRECISION DSEED                                         00033660
INTEGER START,END,POINT1,POINT2,D,FEASLD(20),FEASLN(20)           00033670
INTEGER FTRUCK,FWD,BACK,TEMTRK(120),CUMLD(120),CUMLN(120)        00033680
INTEGER FOUND,TRCNT,TRK,TAG,D1,D2,D3,D4,D5,D6,D7,D8           00033690
INTEGER FSTART,FEND,FPRED(120),FSUCC(120)                         00033700
INTEGER PERMPR(120),PERMSU(120),PERMTR(120)                      00033710
INTEGER DLIMIT,DEPOT(20),LOKK(120)                                00033720
DIMENSION DIST(0:120,0:120),SAVING(3,6000),SORT(6000),PERMSV(40) 00033730
DIMENSION ISORT(6000),JSORT(6000),LOAD(120),TRUCK(120)            00033740
DIMENSION LENGTH(120),WORK(6),TRADE(7,500)                         00033750
COMMON IROUTE,NCITY,IRUN,START,END,WTLIM,DISTLM,ALLOW,NPERM,        00033760
*ICOUNT,LDDVLM,LNDVLM,DLIMIT,MAXLD,MINLD,MAXLN,MINLN,D1,D2,D3,D4, 00033770
*D5,D6,DEPOT,PRED,SUCC,TRUCK,DEMAND,LENGTH,LOAD,CUMLD,CUMLN,TEMTRK, 00033780
*FEASLD,FEASLN,PERMPR,PERMSU,PERMTR,ISORT,JSORT,SORT,DIST,XCOORD, 00033790
*YCOORD,LOKK,TRADE,NTRADE,DSEED                                 00033800
C                                                               00033810
C                                                               00033820
C SET UP HEADERS TO EACH OF NCITY ROUTES.                          00033830
C EACH ROUTE IS INITIALLY ROOTED AT DEPOT.                        00033840
C                                                               00033850
C                                                               00033860
C DO 10 I=1,NCITY                                              00033870
HEAD(I)=I                                                       00033880
TAIL(I)=I                                                       00033890
PRED(I)=0                                                       00033900
SUCC(I)=0                                                       00033910
LOAD(I)=DEMAND(I)                                             00033920
LENGTH(I)=DIST(0,I) + DIST(I,0) + ALLOW                         00033930
10 TRUCK(I)=I                                                 00033940
ROUTES=NCITY                                                 00033950
C                                                               00033960
C                                                               00033970
C START AT TOP OF SAVINGS FILE AND FIND A VALID SAVINGS          00033980
C WHICH CAN BE APPLIED TO A PAIR OF UNLINKED CITIES.             00033990
C                                                               00034000
C                                                               00034010
NSAV=0                                                       00034020
11 NSAV=NSAV+1                                                 00034030
IF(NSAV.GT.ICOUNT) GOTO 15                                     00034040
C                                                               00034050
C ROUTE LENGTH AND WEIGHT LIMITS CANNOT BE EXCEEDED.            00034060
C                                                               00034070
TDIST=FLOAT(LENGTH(TRUCK(ISORT(NSAV)))) +                   00034080

```

```

*      LENGTH(TRUCK(JSORT(NSAV))) - SORT(NSAV) + 0.5          00034090
TWGT=LOAD(TRUCK(ISORT(NSAV))) + LOAD(TRUCK(JSORT(NSAV)))        00034100
IF(TWGT.GT.WTLIM) GOTO 11                                      00034110
IF(TDIST.GT.DISTLM) GOTO 11                                      00034120
C
C      BOTH CITIES TO BE LINKED MUST BE DIRECTLY CONNECTED TO DEPOT. 00034130
C
C      IF(PRED(ISORT(NSAV)).NE.0.AND.SUCC(ISORT(NSAV)).NE.0) GOTO 11 00034150
C      IF(PRED(JSORT(NSAV)).NE.0.AND.SUCC(JSORT(NSAV)).NE.0) GOTO 11 00034160
C
C      BOTH CITIES TO BE LINKED CANNOT BE IN SAME SUBTOUR.           00034170
C
C      IF(TRUCK(ISORT(NSAV)).EQ.TRUCK(JSORT(NSAV))) GOTO 11         00034180
C
C      CASE 1: BOTH CITIES ARE AT BEGINNING OF SUBTOUR.             00034190
C
C      IF(HEAD(TRUCK(ISORT(NSAV))).EQ.ISORT(NSAV).AND.            00034200
*      HEAD(TRUCK(JSORT(NSAV))).EQ.JSORT(NSAV)) THEN            00034210
C
C      HEAD(TRUCK(ISORT(NSAV)))=TAIL(TRUCK(JSORT(NSAV)))          00034220
HEAD(TRUCK(JSORT(NSAV)))=0                                       00034230
TAIL(TRUCK(JSORT(NSAV)))=0                                       00034240
PRED(ISORT(NSAV))=JSORT(NSAV)                                     00034250
LAST=ISORT(NSAV)                                                 00034260
NODE=JSORT(NSAV)                                                 00034270
12    TRUCK(NODE)=TRUCK(ISORT(NSAV))                            00034280
IF(SUCC(NODE).NE.0) THEN                                         00034290
    NEXT=SUCC(NODE)                                              00034300
    PRED(NODE)=NEXT                                             00034310
    SUCC(NODE)=LAST                                            00034320
    LAST=NODE                                                 00034330
    NODE=NEXT                                                 00034340
    GOTO 12                                                 00034350
END IF                                                               00034360
PRED(NODE)=0                                                       00034370
SUCC(NODE)=LAST                                                 00034380
LENGTH(TRUCK(ISORT(NSAV)))=TDIST                                00034390
LOAD(TRUCK(ISORT(NSAV)))=TWGT                                  00034400
ROUTES=ROUTES-1                                               00034410
GOTO 11                                                 00034420
END IF                                                               00034430
C
C      CASE 2: BOTH CITIES ARE AT END OF SUBTOUR.                00034440
C
C      IF(TAIL(TRUCK(ISORT(NSAV))).EQ.ISORT(NSAV).AND.          00034450
*      TAIL(TRUCK(JSORT(NSAV))).EQ.JSORT(NSAV)) THEN           00034460
TAIL(TRUCK(ISORT(NSAV)))=HEAD(TRUCK(JSORT(NSAV)))            00034470
TAIL(TRUCK(JSORT(NSAV)))=0                                     00034480
HEAD(TRUCK(JSORT(NSAV)))=0                                     00034490
SUCC(ISORT(NSAV))=JSORT(NSAV)                                 00034500
LAST=ISORT(NSAV)                                              00034510
NODE=JSORT(NSAV)                                              00034520
13    TRUCK(NODE)=TRUCK(ISORT(NSAV))                            00034530
IF(PRED(NODE).NE.0) THEN                                         00034540
    NEXT=PRED(NODE)                                              00034550
    SUCC(NODE)=NEXT                                             00034560
    PRED(NODE)=LAST                                             00034570
    LAST=NODE                                                 00034580
    NODE=NEXT                                                 00034590
    GOTO 13                                                 00034600
END IF                                                               00034610
SUCC(NODE)=0                                                       00034620
PRED(NODE)=LAST                                                 00034630
LENGTH(TRUCK(ISORT(NSAV)))=TDIST                                00034640
LOAD(TRUCK(ISORT(NSAV)))=TWGT                                  00034650
ROUTES=ROUTES + 1                                              00034660
GOTO 11                                                 00034670
END IF                                                               00034680
C
C      CASE 3: ONE CITY IS AT BEGINNING OF A SUBTOUR,            00034690
C      OTHER IS AT END OF SUBTOUR.                               00034700
C

```

```

C
I=ISORT(NSAV)          00034800
J=JSORT(NSAV)          00034810
IF(HEAD(TRUCK(I)).NE.I) THEN
  I=JSORT(NSAV)
  J=ISORT(NSAV)
END IF
PRED(I)=J              00034820
SUCC(J)=I              00034830
HEAD(TRUCK(I))=HEAD(TRUCK(J))
TAIL(TRUCK(J))=O       00034840
HEAD(TRUCK(J))=O       00034850
NODE=J                 00034860
14 NEXT=PRED(NODE)     00034870
  IF(NEXT.NE.O) THEN
    TRUCK(NODE)=TRUCK(I)
    NODE=NEXT
    GOTO 14
  END IF
  TRUCK(NODE)=TRUCK(I)
  LENGTH(TRUCK(I))=TDIST
  LOAD(TRUCK(I))=TWGT
  ROUTES=ROUTES-1
  GOTO 11
C
C
C
15 IROUTE=O            00034900
  DO 17 I=1,NCITY
    IF(HEAD(I).NE.O) THEN
      IROUTE=IROUTE+1
    END IF
17 CONTINUE
C
C
C
      RENUMBER THE ROUTES, THEN REFORM THE ROUTES INTO SINGLE
      ROUTE WITH ARTIFICIAL DEPOTS INSERTED BETWEEN ADJACENT
      ORIGINAL ROUTES.
C
C
      DO 18 I=NCITY+1,NCITY+IROUTE
        DEMAND(I)=O           00035040
        XCOORD(I)=XCOORD(O)  00035050
18      YCOORD(I)=YCOORD(O)  00035060
      DO 19 I=NCITY+1,NCITY+IROUTE
        DO 19 J=I+1,NCITY+IROUTE
          DIST(I,J)=O         00035070
        DO 19 J=I+1,NCITY+IROUTE
          DIST(J,I)=O         00035080
19      DIST(J,I)=DIST(I,J)  00035090
C
      DO 20 I=1,NCITY
        DO 20 J=NCITY+1,NCITY+IROUTE+1
          DIST(I,J)=DIST(I,O) 00035100
        DIST(J,I)=DIST(I,J)   00035110
20      DIST(J,I)=DIST(I,J)  00035120
C
      I=NCITY
      TRCNT=O               00035130
      DO 21 J=1,NCITY
        IF(HEAD(J).NE.O) THEN
          FOUND=J             00035140
          TRCNT=TRCNT+1
          I=I+1
          PRED(HEAD(J))=I
          SUCC(I)=HEAD(J)
          SUCC(TAIL(J))=I+1
          PRED(I+1)=TAIL(J)
          TRUCK(I)=TRCNT
          DEPOT(TRCNT)=I
        END IF
21      CONTINUE
        SUCC(TAIL(FOUND))=NCITY+1
        PRED(NCITY+1)=TAIL(FOUND)

```

```

C          DO 22 I=1,IROUTE          00035510
C          LENGTH(I)=0           00035520
22       LOAD(I)=0             00035530
C          POINT1=NCITY+1        00035540
C          NODE=POINT1          00035550
23       IF(SUCC(NODE).EQ.NCITY+1) GOTO 24 00035560
C          IF(DEMAND(NODE).EQ.0.) TRK=TRUCK(NODE)
C          TRUCK(NODE)=TRK
C          LENGTH(TRUCK(NODE))=LENGTH(TRUCK(NODE))+DIST(NODE,SUCC(NODE)) 00035570
C          IF(SUCC(NODE).LE.NCITY) LENGTH(TRUCK(NODE))=LENGTH(TRUCK(NODE)) 00035580
C          *                      + ALLOW
C          LOAD(TRUCK(NODE))=LOAD(TRUCK(NODE))+DEMAND(NODE)           00035590
C          NODE=SUCC(NODE)          00035600
C          GOTO 23              00035610
24       CONTINUE               00035620
C          TRUCK(NODE)=TRK
C          LENGTH(TRUCK(NODE))=LENGTH(TRUCK(NODE))+DIST(NODE,SUCC(NODE)) 00035630
C          LOAD(TRUCK(NODE))=LOAD(TRUCK(NODE))+DEMAND(NODE)           00035640
C          RETURN                 00035650
202      FORMAT(1H ,T4,I3)        00035660
C          END
C
C
C*****SUBROUTINE TSP(START,END,PRED,SUCC,LENGTH,DIST,ALLOW)*****00035750
C
C
C          THIS SUBROUTINE IS USED TO SOLVE THE TRAVELING SALESMAN 00035760
C          PROBLEM USING THE 3-OPT EXCHANGE PROCEDURE.                00035770
C
C*****SUBROUTINE TSP(START,END,PRED,SUCC,LENGTH,DIST,ALLOW)*****00035780
C
C
C          CHARACTER*44 IPLACE          00035790
C          INTEGER START,END,PRED(120),SUCC(120),POINT1,POINT2,POINT3 00035800
C          INTEGER DIST(0:120,0:120),D1,D2,D3,D4,D5,D6,FWD,BACK        00035810
C          INTEGER ALLOW, STACK(9),BESTLN,BSTART,BIEND,BESTP(120),BESTS(120) 00035820
C
C          BESTLN=999999            00035830
C          IF(END.EQ.START) THEN    00035840
C          LENGTH=0                00035850
C          RETURN                  00035860
C          ENDIF                   00035870
C          IF(END.EQ.SUCC(START)) THEN 00035880
C          LENGTH=DIST(START,END)+DIST(END,SUCC(END))+ALLOW           00035890
C          RETURN                  00035900
C          END IF                  00035910
C          IF(END.EQ.SUCC(SUCC(START))) THEN 00035920
C          LENGTH=DIST(START,SUCC(START))+DIST(SUCC(START),END)        00035930
C          *                      +DIST(END,SUCC(END))+2*ALLOW           00035940
C          LENGTH=L1+DIST(END,SUCC(END))+2*ALLOW           00035950
C          *                      +2*ALLOW           00035960
C          IF(L1.LE.LENGTH) THEN   00035970
C          LENGTH=L1                00035980
C          RETURN                  00035990
C          ENDIF                   00036000
C          ELSE                     00036010
C          LENGTH=L2                00036020
C          IP1=SUCC(START)          00036030
C          ISUCC=SUCC(END)          00036040
C          SUCC(START)=END          00036050
C          PRED(END)=START          00036060
C          SUCC(END)=IP1            00036070
C          PRED(IP1)=END            00036080
C          SUCC(IP1)=ISUCC          00036090
C          PRED(ISUCC)=IP1          00036100
C          RETURN                  00036110
C          ENDIF                   00036120
C
C          IS=START                 00036130

```

```

IE=END          00036220
IPRED=PRED(START) 00036230
ISUCC=SUCC(END) 00036240
FWD=1          00036250
BACK=2          00036260
PRED(START)=END 00036270
SUCC(END)=START 00036280
C
C
C      BEGIN ITERATIONS
C
C
POINT1=END      00036290
5 POINT1=SUCC(POINT1) 00036300
IF(POINT1.EQ.PRED(PRED(END))) GOTO 100
IS1=SUCC(POINT1) 00036310
POINT2=POINT1 00036320
6 POINT2=SUCC(POINT2) 00036330
IF(POINT2.EQ.PRED(END)) GOTO 5
IS2=SUCC(POINT2) 00036340
POINT3=POINT2 00036350
7 POINT3=SUCC(POINT3) 00036360
IF(POINT3.EQ.END) GOTO 6
IS3=SUCC(POINT3) 00036370
C
C
C      D1=DIST(POINT1,IS1)
D2=DIST(POINT2,IS2) 00036380
D3=DIST(POINT3,IS3) 00036390
C
C
C      3-OPT TYPE I EXCHANGE
C
C
NTYPE=1          00036400
D4=DIST(POINT1,POINT2) 00036410
D5=DIST(SUCC(POINT1),POINT3) 00036420
D6=DIST(SUCC(POINT2),SUCC(POINT3)) 00036430
IF(D1+D2+D3.LE.D4+D5+D6) GOTO 15
C
C
C      ELSE
LAST=POINT1      00036440
NEXT=POINT2      00036450
SUCC(POINT1)=POINT2 00036460
10 NODE=NEXT     00036470
IP=PRED(NODE)    00036480
PRED(NODE)=LAST 00036490
IF(NODE.EQ.IS3) GOTO 82
SUCC(NODE)=IP    00036500
IF(NODE.EQ.IS1) SUCC(NODE)=POINT3 00036510
IF(NODE.EQ.IS2) SUCC(NODE)=IS3 00036520
LAST=NODE        00036530
NEXT=SUCC(NODE)  00036540
GOTO 10          00036550
C
C
C
C
C      3-OPT TYPE II EXCHANGE
C
C
15 NTYPE=2        00036560
D4=DIST(POINT1,IS2) 00036570
D5=DIST(POINT3,POINT2) 00036580

```

```

D6=DIST(IS1,IS3)          00036930
IF(D1+D2+D3.LE.D4+D5+D6) GOTO 25 00036940
C ELSE                      00036950
LAST=POINT1                00036960
NEXT=IS2                   00036970
SUCC(POINT1)=IS2           00036980
MODE=FWD                   00036990
20 NODE=NEXT                00037000
IP=PRED(NODE)              00037010
PRED(NODE)=LAST             00037020
IF(NODE.EQ.IS3) GOTO 82      00037030
IF(MODE.EQ.BACK) SUCC(NODE)=IP 00037040
IF(NODE.EQ.POINT3) THEN     00037050
    SUCC(NODE)=POINT2        00037060
    MODE=BACK                 00037070
END IF                      00037080
IF(NODE.EQ.IS1) SUCC(NODE)=IS3 00037090
LAST=NODE                   00037100
NEXT=SUCC(NODE)              00037110
GOTO 20                      00037120
C
C
C
C   3-OPT TYPE III EXCHANGE
C
C
C
25 NTYPE=3                  00037130
D4=DIST(POINT1,IS2)          00037140
D5=DIST(POINT3,IS1)          00037150
D6=DIST(POINT2,IS3)          00037160
IF(D1+D2+D3.LE.D4+D5+D6) GOTO 35 00037170
C ELSE                      00037180
LAST=POINT1                00037190
NEXT=IS2                   00037200
SUCC(POINT1)=IS2           00037210
30 NODE=NEXT                00037220
PRED(NODE)=LAST              00037230
IF(NODE.EQ.IS3) GOTO 82      00037240
IF(NODE.EQ.POINT3) SUCC(NODE)=IS1 00037250
IF(NODE.EQ.POINT2) SUCC(NODE)=IS3 00037260
LAST=NODE                   00037270
NEXT=SUCC(NODE)              00037280
GOTO 30                      00037290
C
C
C
C   3-OPT TYPE IV EXCHANGE
C
C
C
35 NTYPE=4                  00037300
D4=DIST(POINT1,POINT3)       00037310
D5=DIST(IS2,IS1)             00037320
D6=DIST(POINT2,IS3)          00037330
IF(D1+D2+D3.LE.D4+D5+D6) GOTO 45 00037340
C ELSE                      00037350
LAST=POINT1                00037360
NEXT=POINT3                00037370
SUCC(POINT1)=POINT3         00037380
MODE=BACK                   00037390
40 NODE=NEXT                00037400
IP=PRED(NODE)              00037410
PRED(NODE)=LAST              00037420
IF(NODE.EQ.IS3) GOTO 82      00037430
IF(MODE.EQ.BACK) SUCC(NODE)=IP 00037440
IF(NODE.EQ.IS2) THEN        00037450
    SUCC(NODE)=IS1           00037460
    MODE=FWD                 00037470
END IF                      00037480
IF(NODE.EQ.POINT2) SUCC(NODE)=IS3 00037490

```

```

LAST=NODE          00037640
NEXT=SUCC(NODE)  00037650
GOTO 40          00037660
C
C
C
C      3-OPT TYPE V EXCHANGE
C
C
C
45 NTYPE=5        00037670
D4=DIST(POINT1,POINT2) 00037680
D5=DIST(IS1,IS2)    00037690
IF(D1+D2.LE.D4+D5) GOTO 48 00037700
C     ELSE          00037710
LAST=POINT1       00037720
NEXT=POINT2       00037730
SUCC(POINT1)=POINT2 00037740
00037750
00037760
00037770
00037780
00037790
00037800
00037810
00037820
00037830
00037840
00037850
00037860
00037870
00037880
00037890
00037900
00037910
00037920
00037930
00037940
00037950
00037960
00037970
00037980
00037990
00038000
00038010
00038020
00038030
00038040
00038050
00038060
00038070
00038080
00038090
00038100
00038110
00038120
00038130
00038140
00038150
00038160
00038170
00038180
00038190
00038200
00038210
00038220
00038230
00038240
00038250
00038260
00038270
00038280
00038290
00038300
00038310
00038320
00038330
00038340
46 NODE=NEXT
IP=PRED(NODE)
PRED(NODE)=LAST
IF(NODE.EQ.IS2) GOTO 82
SUCC(NODE)=IP
IF(NODE.EQ.IS1) SUCC(NODE)=IS2
LAST=NODE
NEXT=SUCC(NODE)
GOTO 46
C
C
C
C      3-OPT TYPE VI EXCHANGE
C
C
C
48 NTYPE=6        00037940
D4=DIST(POINT2,POINT3) 00037950
D5=DIST(IS2,IS3)    00037960
IF(D2+D3.LE.D4+D5) GOTO 54 00037970
LAST=POINT2       00037980
NEXT=POINT3       00037990
SUCC(POINT2)=POINT3 00038000
00038010
00038020
00038030
00038040
00038050
00038060
00038070
00038080
00038090
00038100
00038110
00038120
00038130
00038140
00038150
00038160
00038170
00038180
00038190
00038200
00038210
00038220
00038230
00038240
00038250
00038260
00038270
00038280
00038290
00038300
00038310
00038320
00038330
00038340
50 NODE=NEXT
IP=PRED(NODE)
PRED(NODE)=LAST
IF(NODE.EQ.IS3) GOTO 82
SUCC(NODE)=IP
IF(NODE.EQ.IS2) SUCC(NODE)=IS3
LAST=NODE
NEXT=SUCC(NODE)
GOTO 50
C
C
C
C      3-OPT TYPE VII EXCHANGE
C
C
C
54 NTYPE=7        00038210
D4=DIST(POINT1,POINT3) 00038220
D5=DIST(IS1,IS3)    00038230
IF(D1+D3.LE.D4+D5) GOTO 7 00038240
LAST=POINT1       00038250
NEXT=POINT3       00038260
SUCC(POINT1)=POINT3 00038270
00038280
00038290
00038300
00038310
00038320
00038330
00038340
55 NODE=NEXT
IP=PRED(NODE)
PRED(NODE)=LAST
IF(NODE.EQ.IS3) GOTO 82
SUCC(NODE)=IP
IF(NODE.EQ.IS1) SUCC(NODE)=IS3
LAST=NODE

```

```

NEXT=SUCC(NODE)          00038350
GOTO 55                 00038360
C
C
C      ROTATE             00038370
C
C
C
B2 FSTART=END           00038380
END=PRED(END)            00038390
START=FSTART              00038400
POINT1=END                00038410
GOTO 5                   00038420
C
C
C
C      CALCULATE LENGTH OF ROUTE 00038430
C
C
C
C
100 START=IS             00038440
IEND=PRED(START)          00038450
END=IEND                  00038460
LN=DIST(IEND,ISUCC)       00038470
NEXT=SUCC(START)           00038480
101 NODE=NEXT             00038490
LN=LN+DIST(NODE,PRED(NODE))+ALLOW 00038500
IF(NODE.EQ.IEND) GOTO 102  00038510
NEXT=SUCC(NODE)            00038520
GOTO 101                 00038530
C
C
C
C      STORE THE BEST TSP SOLUTION 00038540
C
C
C
C
102 IF(LN.GE.BESTLN) GOTO 104  00038550
C      ELSE                  00038560
BESTLN=LN                  00038570
BSTART=START                00038580
BIEND=IEND                  00038590
BESTS(START)=SUCC(START)    00038600
BESTP(START)=IEND            00038610
NEXT=SUCC(START)             00038620
103 NODE=NEXT               00038630
BESTP(NODE)=PRED(NODE)       00038640
BESTS(NODE)=SUCC(NODE)       00038650
IF(NODE.EQ.IEND) GOTO 104   00038660
NEXT=SUCC(NODE)              00038670
GOTO 103                 00038680
C
C
C
C      ALTER ROUTE STRUCTURE FOR NEXT TSP SOLUTION 00038690
C
C
C
C
104 NTSP=NTSP+1             00038700
IF(NTSP.EQ.1) LN1=LN         00038710
IF(NTSP.EQ.2) LN2=LN         00038720
IF(NTSP.LT.2) THEN          00038730
  I1=SUCC(START)             00038740
  I2=PRED(END)               00038750
  I3=PRED(I2)                 00038760
  SUCC(START)=I2              00038770
  PRED(I2)=START              00038780
  SUCC(I2)=I1                 00038790
  PRED(I1)=I2                 00038800
  SUCC(I3)=END                00038810
  PRED(END)=I3                 00038820
  POINT1=END                  00038830
  GOTO 5                      00038840
C
C
C
C

```

```

ENDIF                                         00039060
C                                             00039070
C                                             00039080
C                                             00039090
C RETRIEVE BEST TSP SOLUTION                 00039100
C                                             00039110
C                                             00039120
C LENGTH=BESTLN                                00039130
C START=BESTSTART                             00039140
C SUCC(START)=BESTS(START)                     00039150
C IEND=BIEND                                 00039160
C PRED(START)=BIEND                           00039170
C NEXT=BESTS(START)                           00039180
105 NODE=NEXT                                 00039190
C PRED(NODE)=BESTP(NODE)                      00039200
C SUCC(NODE)=BESTS(NODE)                      00039210
C IF(NODE.EQ.IEND) GOTO 106                   00039220
C NEXT=BESTS(NODE)                           00039230
C GOTO 105                                    00039240
C                                             00039250
C                                             00039260
C                                             00039270
C RE-ESTABLISH ROUTE'S POSITION WITHIN OVERALL ROUTE STRUCTURE 00039280
C                                             00039290
C                                             00039300
C 106 START=IS                                00039310
C IEND=PRED(START)                           00039320
C SUCC(IEND)=ISUCC                          00039330
C PRED(ISUCC)=IEND                           00039340
C PRED(START)=IPRED                          00039350
C SUCC(IPRED)=START                         00039360
C RETURN                                     00039370
C END                                         00039380
C                                             00039390
C                                             00039400
C*****                                         00039410
C                                             00039420
C SUBROUTINE FEAS2(P1,P2,FEAS,NTYPE,DIFF,DSTRXL) 00039430
C                                             00039440
C                                             00039450
C THIS SUBROUTINE DETERMINES THE FEASIBILITY OF A 2-OPT EXCHANGE. 00039460
C                                             00039470
C*****                                         00039480
C                                             00039490
C                                             00039500
C CHARACTER*44 PNAME,IPLACE                    00039510
C INTEGER FMAXLD,FMINLD,FMAXLN,FMINLN,FEAS,P1,P2,DIFF          00039520
C INTEGER EUCLID,CITY,XCOORD(0:120),YCOORD(0:120),DEMAND(0:120) 00039530
C INTEGER HEAD(120),TAIL(120),PRED(120),SUCC(120),ROUTES,TWGT,WTLIM 00039540
C INTEGER DIST,ALLOW,TDIST,DISTLM,TRUCK,IR(100),IFLAG(40),        00039550
* PERMI(40),PERMJ(40)                         00039560
DOUBLE PRECISION DSEED                      00039570
C INTEGER START,END,POINT1,D,FEASLD(20),FEASLN(20)           00039580
C INTEGER FTRUCK,FWD,BACK,TEMTRK(120),CUMLD(120),CUMLN(120) 00039590
C INTEGER FOUND,TRCNT,TRK,TAG,D1,D2,D3,D4,D5,D6,D7,D8,DEES(6) 00039600
C INTEGER FSTART,FEND,FPRED(120),FSUCC(120)                  00039610
C INTEGER PERMPR(120),PERMSU(120),PERMTR(120)                00039620
C INTEGER DLIMIT,DEPOT(20),DSTRXL                 00039630
C DIMENSION DIST(0:120,0:120),SAVING(3,6000),SORT(6000),PERMSV(40) 00039640
C DIMENSION ISORT(6000),JSORT(6000),LOAD(120),TRUCK(120)       00039650
C DIMENSION LENGTH(120),WORK(6),LOKK(120),TRADE(7,500)         00039660
COMMON IROUTE,NCITY,IRUN,START,END,WTLIM,DISTLM,ALLOW,NPERM,    00039670
*ICOUNT,LDDVLM,LNDVLM,DLIMIT,MAXLD,MINLD,MAXLN,MINLN,DEES,      00039680
*DEPOT,PRED,SUCC,TRUCK,DEMAND,LENGTH,LOAD,CUMLD,CUMLN,TEMTRK, 00039690
*FEASLD,FEASLN,PERMPR,PERMSU,PERMTR,ISORT,JSORT,SORT,DIST,XCOORD, 00039700
*YCOORD,LOKK,TRADE,NTRADE,DSEED                   00039710
C                                             00039720
C                                             00039730
C FEAS=0                                         00039740
C LDDEV=MAXLD-MINLD                           00039750
C LNDEV=MAXLN-MINLN                           00039760

```

```

C                                         00039770
C                                         00039780
C                                         00039790
C IF BOTH POINTS ARE IN SAME ROUTE, THE EXCHANGE IS FEASIBLE. 00039800
C                                         00039810
C                                         00039820
C                                         00039830
C                                         00039840
C                                         00039850
C                                         00039860
C                                         00039870
C                                         00039880
C                                         00039890
C                                         00039900
C                                         00039910
C                                         00039920
C                                         00039930
C                                         00039940
C                                         00039950
C                                         00039960
C                                         00039970
C                                         00039980
C                                         00039990
C                                         00040000
C                                         00040010
C                                         00040020
C IF POINTS P1 AND P2 ARE IN DIFFERENT ROUTES, BOTH ROUTES MIGHT 00040030
C BE AFFECTED BY AN EXCHANGE. 00040040
C                                         00040050
C                                         00040060
C                                         00040070
C                                         00040080
C                                         00040090
C                                         00040100
C                                         00040110
C                                         00040120
C                                         00040130
C                                         00040140
C                                         00040150
C                                         00040160
C                                         00040170
C                                         00040180
C                                         00040190
C                                         00040200
C                                         00040210
C                                         00040220
C                                         00040230
C                                         00040240
C                                         00040250
C                                         00040260
C                                         00040270
C                                         00040280
C                                         00040290
C ROUTE LENGTH DEVIATION AND ROUTE LOAD DEVIATION TEST. 00040300
C                                         00040310
C                                         00040320
C                                         00040330
C                                         00040340
C                                         00040350
C                                         00040360
C                                         00040370
C                                         00040380
C                                         00040390
C                                         00040400
C                                         00040410
C                                         00040420
C                                         00040430
C                                         00040440
C                                         00040450
C                                         00040460
C                                         00040470

```

```

      IF(LOKK(DEPOT(I)).NE.0) GOTO 5          00040480
      IF(FEASLD(I).GT.FMAXLD) FMAXLD=FEASLD(I) 00040490
      IF(FEASLD(I).LT.FMINLD.AND.FEASLD(I).NE.0) FMINLD=FEASLD(I) 00040500
  5 CONTINUE                                00040510
C                                         00040520
C                                         00040530
C                                         00040540
C TRADEOFF ANALYSIS                         00040550
C                                         00040560
C                                         00040570
IF(DSTRXL.GT.0) GOTO 6          00040580
IF(NTYPE.EQ.0) GOTO 6          00040590
IF(FMAXLD-FMINLD.GT.LDDVLM.OR.FMAXLN-FMINLN.GT.LNDVLM) THEN 00040600
  NTRADE=NTRADE+1                  00040610
  IF(NTRADE.GT.500) NTRADE=500        00040620
  TRADE(1,NTRADE)=-DIFF           00040630
  TRADE(2,NTRADE)=FMAXLD-FMINLD-LDDEV 00040640
  TRADE(3,NTRADE)=FMAXLN-FMINLN-LNDEV 00040650
  TRADE(4,NTRADE)=P1              00040660
  TRADE(5,NTRADE)=P2              00040670
  TRADE(6,NTRADE)=0.              00040680
  TRADE(7,NTRADE)=0.              00040690
ENDIF                                    00040700
  6 IF(FMAXLN-FMINLN.GT.LNDVLM) RETURN 00040710
    IF(FMAXLD-FMINLD.GT.LDDVLM) RETURN 00040720
C                                         00040730
C                                         00040740
C                                         00040750
C                                         00040760
C                                         00040770
C                                         00040780
C ELSE                                 00040790
C EXCHANGE IS FEASIBLE.                 00040800
C                                         00040810
  FEAS=1                               00040820
  MAXLD=FMAXLD                         00040830
  MINLD=FMINLD                         00040840
  MAXLN=FMAXLN                         00040850
  MINLN=FMINLN                         00040860
  RETURN                                00040870
  END                                   00040880
C                                         00040890
C                                         00040900
C                                         00040910
C                                         00040920
C*****                                         00040930
C                                         00040940
SUBROUTINE LNDV2(LNRLX)          00040950
C                                         00040960
C                                         00040970
C THIS SUBROUTINE IS USED TO MINIMIZE THE MAXIMUM LENGTH DEVIATION 00040980
C IN ROUTE LENGTHS VIA THE TWO-ARC BRANCH EXCHANGE METHOD.       00040990
C                                         00041000
C*****                                         00041010
C                                         00041020
C                                         00041030
CHARACTER*1 MODE                00041040
CHARACTER*44 PNAME               00041050
DOUBLE PRECISION DSEED          00041060
INTEGER START,END,WTLIM,DISTLM,ALLOW,DLIMIT,D1,D2,D3,D4,D5,D6, 00041070
*      DEPOT(20),PRED(120),SUCC(120),FPRED(120),FSUCC(120), 00041080
*      TRUCK(120),DEMAND(0:120),LENGTH(120),LOAD(120),CUMLD(120), 00041090
*      CUMLN(120),TEMTRK(120),FEASLD(20),FEASLN(20),PERMPR(120), 00041100
*      PERMSU(120),PERMTR(120),ISORT(6000),JSORT(6000), 00041110
*      DIST(0:120,0:120),XCOORD(0:120),YCOORD(0:120),TLOAD,TDIST, 00041120
*      POINT1,POINT2,FMAXLD,FMINLD,FMAXLN,FMINLN,DEPOT4,TWTLM 00041130
DIMENSION SORT(6000),LOKK(120),TRADE(7,500) 00041140
COMMON IROUTE,NCITY,IRUN,START,END,WTLIM,DISTLM,ALLOW,NPERM, 00041150
*ICOUNT,LDDVLM,LNDVLM,DLIMIT,MAXLD,MINLD,MAXLN,MINLN,D1,D2,D3,D4, 00041160
*D5,D6,DEPOT,PRED,SUCC,TRUCK,DEMAND,LENGTH,LOAD,CUMLD,CUMLN,TEMTRK, 00041170
*FEASLD,FEASLN,PERMPR,PERMSU,PERMTR,ISORT,JSORT,SORT,DIST,XCOORD, 00041180

```

```

*YCOORD,LOKK,TRADE,NTRADE,DSEED          00041190
C                                         00041200
C                                         00041210
C                                         00041220
C                                         00041230
C                                         00041240
TLOAD=0                                     00041250
TDIST=0                                     00041260
START=GGUBFS(DSEED)*NCITY+1                 00041270
END=PRED(START)                            00041280
TWTLIM=MINO(WTLIM,MAXLD+LDDVLM)           00041290
DO 1 I=1,IRROUTE                         00041300
    FEASLD(I)=LOAD(I)                      00041310
    FEASLN(I)=LENGTH(I)                   00041320
    TDIST=TDIST+LENGTH(I)                00041330
1   TLOAD=TLOAD+LOAD(I)                    00041340
    DO 2 I=1,NCITY+IRROUTE               00041350
        FPRED(I)=PRED(I)                  00041360
2   FSUCC(I)=SUCC(I)                     00041370
C                                         00041380
C                                         00041390
C DETERMINE LOAD AND LENGTH DEVIATIONS. 00041400
C                                         00041410
C                                         00041420
LDDEV=MAXLD-MINLD                         00041430
LNDEV=MAXLN-MINLN                         00041440
C                                         00041450
C                                         00041460
C BEGIN ITERATIONS                         00041470
C                                         00041480
C                                         00041490
C                                         00041500
POINT1=PRED(START)                       00041510
6  POINT1=SUCC(POINT1)                   00041520
    IF(POINT1.EQ.PRED(PRED(END))) THEN    00041530
        RETURN                                00041540
    ENDIF                                 00041550
    MODE='F'                               00041560
    POINT2=SUCC(POINT1)                   00041570
7  POINT2=SUCC(POINT2)                   00041580
    IF(POINT2.EQ.END) GOTO 6             00041590
    IF(LOKK(POINT2).EQ.1) GOTO 7         00041600
    IF(TRUCK(POINT1).NE.TRUCK(POINT2)) MODE='B'
C                                         00041610
C                                         00041620
C                                         00041630
C                                         00041640
C DETERMINE WHETHER BOTH ARCS ARE IN THE SAME ROUTE. IF SO, IGNORE 00041650
C THIS EXCHANGE.                          00041660
C                                         00041670
C                                         00041680
C                                         00041690
C                                         00041700
C                                         00041710
C                                         00041720
C DETERMINE WHETHER AT LEAST ONE OF THE ARCS IS IN THE LONG ROUTE 00041730
C OR IN THE SHORT ROUTE. IF NOT, IGNORE THE EXCHANGE.            00041740
C                                         00041750
C                                         00041760
C                                         00041770
C                                         00041780
*   GOTO 7                                00041790
    IF(LENGTH(TRUCK(POINT1)).EQ.MAXLN) GOTO 8
    IF(LENGTH(TRUCK(POINT1)).EQ.MINLN) GOTO 8
    IF(LENGTH(TRUCK(POINT2)).EQ.MAXLN) GOTO 8
    IF(LENGTH(TRUCK(POINT2)).EQ.MINLN) GOTO 8
C   ELSE                                00041830
C   GOTO 7                                00041840
C                                         00041850
C                                         00041860
C                                         00041870
C LOAD FEASIBILITY TEST                  00041880
C                                         00041890

```

```

C
 8 IF(CUMLD(POINT1)+CUMLD(POINT2).GT.TWTL) GOTO 7          00041900
    IF(LOAD(TRUCK(POINT1))-CUMLD(POINT1)+LOAD(TRUCK(POINT2))- 00041910
     * CUMLD(POINT2).GT.TWTL) GOTO 7                      00041920
    FEASLD(TRUCK(POINT1))=CUMLD(POINT1)+CUMLD(POINT2)      00041930
    FEASLD(TRUCK(POINT2))=LOAD(TRUCK(POINT1))-CUMLD(POINT1)+ 00041940
     * LOAD(TRUCK(POINT2))-CUMLD(POINT2)                  00041950
00041960
C
C
C
C   DETERMINE WHETHER LOAD DEVIATION LIMITS HAVE BEEN MAINTAINED. 00041970
C
C
C
C   FMAXLD=-99          00041980
C   FMINLD=999999         00041990
DO 9 I=1,IROUTE
  IF(LOKK(DEPOT(I)).NE.0) GOTO 9          00042000
  IF(FEASLD(I).GT.FMAXLD) FMAXLD=FEASLD(I) 00042010
  IF(FEASLD(I).LT.FMINLD.AND.FEASLD(I).NE.0) FMINLD=FEASLD(I) 00042020
9 CONTINUE
  IF(FMAXLD-FMINLD.GT.LDDVLM) THEN          00042030
    FEASLD(TRUCK(POINT1))=LOAD(TRUCK(POINT1)) 00042040
    FEASLD(TRUCK(POINT2))=LOAD(TRUCK(POINT2)) 00042050
    GOTO 7                                     00042060
  ENDIF
C
C
C
C   ESTABLISH POTENTIAL ROUTE STRUCTURE RESULTING FROM EXCHANGE. 00042070
C
C
C
C   CALL FXCH2(POINT1,POINT2,FPRED,FSUCC,NULL)           00042080
  IF(NULL.EQ.1) THEN                         00042090
    FEASLD(TRUCK(POINT1))=LOAD(TRUCK(POINT1)) 00042100
    FEASLD(TRUCK(POINT2))=LOAD(TRUCK(POINT2)) 00042110
    GOTO 7                                     00042120
  ENDIF
C
C
C
C   SOLVE TSP FOR EACH ROUTE CHANGED BY THE ARC EXCHANGE. 00042130
C
C
C
C   ISTR=DEPOT(TRUCK(POINT1))          00042140
  NEXT=FSUCC(ISTR)                         00042150
10 NODE=NEXT
  IF(NODE.GT.NCITY) THEN          00042160
    IEND=FPRED(NODE)
    CALL TSP(ISTR,IEND,FPRED,FSUCC,LNGTH,DIST,ALLOW) 00042170
    NTSP=NTSP+1
    IF(LNGTH.GT.DISTLM) GOTO 15          00042180
    FEASLN(TRUCK(POINT1))=LNGTH          00042190
    GOTO 11
  ENDIF
  NEXT=FSUCC(NODE)                         00042200
  GOTO 10
11 ISTR=DEPOT(TRUCK(POINT2))          00042210
  NEXT=FSUCC(ISTR)                         00042220
12 NODE=NEXT
  IF(NODE.GT.NCITY) THEN          00042230
    IEND=FPRED(NODE)
    CALL TSP(ISTR,IEND,FPRED,FSUCC,LNGTH,DIST,ALLOW) 00042240
    NTSP=NTSP+1
    IF(LNGTH.GT.DISTLM) GOTO 15          00042250
    FEASLN(TRUCK(POINT2))=LNGTH          00042260
    GOTO 13
  ENDIF
  NEXT=FSUCC(NODE)                         00042270
  GOTO 12
C
C

```

```

C          DETERMINE WHETHER DISTANCE INCREASE IS WITHIN ACCEPTABLE LIMITS.      00042610
C          DETERMINE WHETHER DISTANCE INCREASE IS WITHIN ACCEPTABLE LIMITS.      00042620
C          DETERMINE WHETHER DISTANCE INCREASE IS WITHIN ACCEPTABLE LIMITS.      00042630
C          DETERMINE WHETHER DISTANCE INCREASE IS WITHIN ACCEPTABLE LIMITS.      00042640
C          DETERMINE WHETHER DISTANCE INCREASE IS WITHIN ACCEPTABLE LIMITS.      00042650
C          DETERMINE WHETHER DISTANCE INCREASE IS WITHIN ACCEPTABLE LIMITS.      00042660
C          DETERMINE WHETHER DISTANCE INCREASE IS WITHIN ACCEPTABLE LIMITS.      00042670
C          DETERMINE WHETHER DISTANCE INCREASE IS WITHIN ACCEPTABLE LIMITS.      00042680
C          DETERMINE WHETHER DISTANCE INCREASE IS WITHIN ACCEPTABLE LIMITS.      00042690
C          DETERMINE WHETHER DISTANCE INCREASE IS WITHIN ACCEPTABLE LIMITS.      00042700
C          DETERMINE WHETHER DISTANCE INCREASE IS WITHIN ACCEPTABLE LIMITS.      00042710
C          DETERMINE WHETHER DISTANCE INCREASE IS WITHIN ACCEPTABLE LIMITS.      00042720
C          DETERMINE WHETHER DISTANCE INCREASE IS WITHIN ACCEPTABLE LIMITS.      00042730
C          DETERMINE WHETHER DISTANCE INCREASE IS WITHIN ACCEPTABLE LIMITS.      00042740
C          DETERMINE WHETHER DISTANCE INCREASE IS WITHIN ACCEPTABLE LIMITS.      00042750
C          DETERMINE WHETHER DISTANCE INCREASE IS WITHIN ACCEPTABLE LIMITS.      00042760
C          DETERMINE WHETHER DISTANCE INCREASE IS WITHIN ACCEPTABLE LIMITS.      00042770
C          DETERMINE WHETHER DISTANCE INCREASE IS WITHIN ACCEPTABLE LIMITS.      00042780
C          DETERMINE WHETHER DISTANCE INCREASE IS WITHIN ACCEPTABLE LIMITS.      00042790
C          DETERMINE WHETHER DISTANCE INCREASE IS WITHIN ACCEPTABLE LIMITS.      00042800
C          DETERMINE WHETHER DISTANCE INCREASE IS WITHIN ACCEPTABLE LIMITS.      00042810
C          DETERMINE WHETHER DISTANCE INCREASE IS WITHIN ACCEPTABLE LIMITS.      00042820
C          DETERMINE WHETHER DISTANCE INCREASE IS WITHIN ACCEPTABLE LIMITS.      00042830
C          DETERMINE WHETHER DISTANCE INCREASE IS WITHIN ACCEPTABLE LIMITS.      00042840
C          DETERMINE WHETHER DISTANCE INCREASE IS WITHIN ACCEPTABLE LIMITS.      00042850
C          DETERMINE WHETHER DISTANCE INCREASE IS WITHIN ACCEPTABLE LIMITS.      00042860
C          DETERMINE WHETHER DISTANCE INCREASE IS WITHIN ACCEPTABLE LIMITS.      00042870
C          DETERMINE WHETHER DISTANCE INCREASE IS WITHIN ACCEPTABLE LIMITS.      00042880
C          DETERMINE WHETHER DISTANCE INCREASE IS WITHIN ACCEPTABLE LIMITS.      00042890
C          DETERMINE WHETHER DISTANCE INCREASE IS WITHIN ACCEPTABLE LIMITS.      00042900
C          DETERMINE WHETHER ROUTE LENGTH DEVIATION IS REDUCED.      00042910
C          DETERMINE WHETHER ROUTE LENGTH DEVIATION IS REDUCED.      00042920
C          DETERMINE WHETHER ROUTE LENGTH DEVIATION IS REDUCED.      00042930
C          DETERMINE WHETHER ROUTE LENGTH DEVIATION IS REDUCED.      00042940
C          DETERMINE WHETHER ROUTE LENGTH DEVIATION IS REDUCED.      00042950
C          DETERMINE WHETHER ROUTE LENGTH DEVIATION IS REDUCED.      00042960
C          DETERMINE WHETHER ROUTE LENGTH DEVIATION IS REDUCED.      00042970
C          DETERMINE WHETHER ROUTE LENGTH DEVIATION IS REDUCED.      00042980
C          DETERMINE WHETHER ROUTE LENGTH DEVIATION IS REDUCED.      00042990
C          CONTINUE      00043000
C          IF(FMAXLN-FMINLN.GE.LNDEV+LNRLX) GOTO 15      00043010
C          ELSE      00043020
C          :      00043030
C          :      00043040
C          :      00043050
C          CHANGE ROUTE STRUCTURE.      00043060
C          :      00043070
C          :      00043080
C          MAXLN=FMAXLN      00043090
C          MINLN=FMINLN      00043100
C          MAXLD=FMAXLD      00043110
C          MINLD=FMINLD      00043120
C          LNDEV=MAXLN-MINLN      00043130
C          LDDEV=MAXLD-MINLD      00043140
C          DO 19 I=1,NCITY+IROUTE      00043150
C          PRED(I)=FPRED(I)      00043160
C          SUCC(I)=FSUCC(I)      00043170
C          NEXT=NCITY+1      00043180
C          INSIDE=0      00043190
C          NODE=NEXT      00043200
C          IF(NODE.EQ.NCITY+1.AND.INSIDE.EQ.1) GOTO 21      00043210
C          INSIDE=1      00043220
C          IF(NODE.GT.NCITY) THEN      00043230
C          ITRK=TRUCK(NODE)      00043240
C          ILD=0      00043250
C          ILN=0      00043260
C          ENDIF      00043270
C          ILD=ILD+DEMAND(NODE)      00043280
C          IF(NODE.LE.NCITY) ILN=ILN+DIST(NODE,PRED(NODE))+ALLOW      00043290
C          CUMLD(NODE)=ILD      00043300
C          CUMLN(NODE)=ILN      00043310

```

```

TRUCK(NODE)=ITRK          00043320
FEASLD(ITRK)=ILD         00043330
NEXT=SUC(NODE)            00043340
IF(NEXT.GT.NCITY) FEASLN(ITRK)=ILN+DIST(NODE,NEXT) 00043350
GOTO 20                  00043360
21 TDIST=0                00043370
TLOAD=0                  00043380
DO 22 I=1,IROUTE          00043390
  LOAD(I)=FEASLD(I)       00043400
  LENGTH(I)=FEASLN(I)    00043410
  TLOAD=TLOAD+LOAD(I)    00043420
22 TDIST=TDIST+LENGTH(I) 00043430
NEXT=NCITY+1              00043440
INSIDE=0                 00043450
C                         00043460
C                         00043470
C                         00043480
C   ROTATE                00043490
C                         00043500
C                         00043510
TWTLIM=MINO(WTLIM,MAXLD+LDDVLM) 00043520
LNRLX=0                  00043530
ISTART=END                00043540
END=PRED(END)             00043550
START=ISTART              00043560
POINT1=END                00043570
GOTO 6                   00043580
END                      00043590
C                         00043600
C                         00043610
C                         00043620
C                         00043630
C                         00043640
C*****00043650
C   SUBROUTINE LDDV2(LDRLX) 00043660
C                         00043670
C                         00043680
C                         00043690
C   THIS SUBROUTINE IS USED TO MINIMIZE THE MAXIMUM LOAD DEVIATION 00043700
C   IN ROUTE LOADS VIA THE TWO-ARC BRANCH EXCHANGE METHOD.        00043710
C                         00043720
C*****00043730
C                         00043740
C                         00043750
CHARACTER*1 MODE          00043760
CHARACTER*44 PNAME         00043770
DOUBLE PRECISION DSEED     00043780
INTEGER START,END,WTLIM,DISTLM,ALLOW,DLIMIT,D1,D2,D3,D4,D5,D6, 00043790
*      DEPOT(20),PRED(120),SUCC(120),FPRED(120),FSUCC(120), 00043800
*      TRUCK(120),DEMAND(0:120),LENGTH(120),LOAD(120),CUMLD(120), 00043810
*      CUMLN(120),TEMTRK(120),FEASLD(20),FEASLN(20),PERMPR(120), 00043820
*      PERMSU(120),PERMTR(120),ISORT(6000),JSORT(6000), 00043830
*      DIST(0:120,0:120),XCOORD(0:120),YCOORD(0:120),TLOAD,TDIST, 00043840
*      POINT1,POINT2,FMAXLD,FMINLD,FMAXLN,FMINLN,DEPOT4,TWTLIM 00043850
DIMENSION SORT(6000),LOKK(120),TRADE(7,500) 00043860
COMMON IROUTE,NCITY,IRUN,START,END,WTLIM,DISTLM,ALLOW,NPERM, 00043870
*ICOUNT,LDDVLM,LNDVLM,DLIMIT,MAXLD,MINLD,MAXLN,MINLN,D1,D2,D3,D4, 00043880
*D5,D6,DEPOT,PRED,SUCC,TRUCK,DEMAND,LENGTH,LOAD,CUMLD,CUMLN,TEMTRK, 00043890
*FEASLD,FEASLN,PERMPR,PERMSU,PERMTR,ISORT,JSORT,SORT,DIST,XCOORD, 00043900
*YCOORD,LOKK,TRADE,NTRADE,DSEED 00043910
C                         00043920
C                         00043930
C                         00043940
C                         00043950
TLOAD=0                  00043960
TDIST=0                  00043970
NTSP=0                   00043980
START=GGUBFS(DSEED)*NCITY+1 00043990
END=PRED(START)           00044000
DO 1 I=1,IROUTE           00044010
  FEASLD(I)=LOAD(I)       00044020

```

```

        FEASLN(I)=LENGTH(I)          00044030
        TDIST=TDIST+LENGTH(I)       00044040
1      TLOAD=TLOAD+LOAD(I)        00044050
      DO 2 I=1,NCITY+IROUTE      00044060
        FPRED(I)=PRED(I)         00044070
2      FSUCC(I)=SUCC(I)         00044080
C
C
C
C      DETERMINE LOAD AND LENGTH DEVIATIONS.    00044090
C
C      LNDEV=MAXLN-MINLN        00044100
      LDDEV=MAXLD-MINLD        00044110
C
C
C      BEGIN ITERATIONS        00044120
C
C
C      POINT1=PRED(START)       00044130
6     POINT1=SUCC(POINT1)       00044140
      IF(POINT1.EQ.PRED(PRED(END))) THEN   00044150
        RETURN                   00044160
      ENDIF                     00044170
      IF(LOKK(POINT1).EQ.1) GOTO 6       00044180
      MODE='F'
      POINT2=SUCC(POINT1)           00044190
7     POINT2=SUCC(POINT2)           00044200
      IF(POINT2.EQ.END) GOTO 6       00044210
      IF(LOKK(POINT2).EQ.1) GOTO 7       00044220
      IF(TRUCK(POINT1).NE.TRUCK(POINT2)) MODE='B'
C
C
C
C      DETERMINE WHETHER BOTH ARCS ARE IN THE SAME ROUTE. IF SO, IGNORE 00044230
C      THIS EXCHANGE.             00044240
C
C
C      IF(TRUCK(POINT1).EQ.TRUCK(POINT2)) GOTO 7       00044250
C
C
C
C      DETERMINE WHETHER AT LEAST ONE OF THE ARCS IS IN THE HEAVY ROUTE 00044260
C      OR IN THE LIGHT ROUTE. IF NOT, IGNORE THE EXCHANGE.            00044270
C
C
C      IF(LOAD(TRUCK(POINT1)).LE.0.OR.LOAD(TRUCK(POINT2)).LE.0)       00044280
*      GOTO 7                  00044290
      IF(LOAD(TRUCK(POINT1)).EQ.MAXLD) GOTO 8       00044300
      IF(LOAD(TRUCK(POINT1)).EQ.MINLD) GOTO 8       00044310
      IF(LOAD(TRUCK(POINT2)).EQ.MAXLD) GOTO 8       00044320
      IF(LOAD(TRUCK(POINT2)).EQ.MINLD) GOTO 8       00044330
C      ELSE                      00044340
      GOTO 7                  00044350
C
C
C
C      LOAD FEASIBILITY TEST    00044360
C
C
C
C      8 IF(CUMLD(POINT1)+CUMLD(POINT2).GT.TWTLIM) GOTO 7       00044370
      IF(LOAD(TRUCK(POINT1))-CUMLD(POINT1)+LOAD(TRUCK(POINT2))- 00044380
*      CUMLD(POINT2).GT.TWTLIM) GOTO 7       00044390
      FEASLD(TRUCK(POINT1))=CUMLD(POINT1)+CUMLD(POINT2)       00044400
      FEASLD(TRUCK(POINT2))=LOAD(TRUCK(POINT1))-CUMLD(POINT1)+ 00044410
*              LOAD(TRUCK(POINT2))-CUMLD(POINT2)       00044420
C
C
C      DETERMINE WHETHER LOAD DEVIATION IS REDUCED.        00044430
C

```

```

C          00044740
FMAXLD=-99 00044750
FMINLD=999999 00044760
DO 9 I=1,IROUTE 00044770
  IF(LOKK(DEPOT(I)).NE.0) GOTO 9 00044780
  IF(FEASLD(I).GT.FMAXLD) FMAXLD=FEASLD(I) 00044790
  IF(FEASLD(I).LT.FMINLD.AND.FEASLD(I).NE.0) FMINLD=FEASLD(I) 00044800
9 CONTINUE 00044810
IF(FMAXLD-FMINLD.GE.LDDEV+LDRLX) THEN 00044820
  FEASLD(TRUCK(POINT1))=LOAD(TRUCK(POINT1)) 00044830
  FEASLD(TRUCK(POINT2))=LOAD(TRUCK(POINT2)) 00044840
  GOTO 7 00044850
ENDIF 00044860
C          00044870
C          00044880
C          00044890
C          ESTABLISH POTENTIAL ROUTE STRUCTURE RESULTING FROM EXCHANGE. 00044900
C          00044910
C          00044920
C          CALL FXCH2(POINT1,POINT2,FPRED,FSUCC,NULL) 00044930
IF(NULL.EQ.1) THEN 00044940
  FEASLD(TRUCK(POINT1))=LOAD(TRUCK(POINT1)) 00044950
  FEASLD(TRUCK(POINT2))=LOAD(TRUCK(POINT2)) 00044960
  GOTO 7 00044970
ENDIF 00044980
C          00044990
C          00045000
C          00045010
C          SOLVE TSP FOR EACH ROUTE IN THE ARC EXCHANGE. 00045020
C          00045030
C          00045040
C          ISTRT=DEPOT(TRUCK(POINT1)) 00045050
NEXT=FSUCC(ISTRT) 00045060
10 NODE=NEXT 00045070
  IF(NODE.GT.NCITY) THEN 00045080
    IEND=FPRED(NODE) 00045090
    CALL TSP(ISTRT,IEND,FPRED,FSUCC,LNGTH,DIST,ALLOW) 00045100
    IF(LNGTH.GT.DISTLM) GOTO 15 00045110
    FEASLN(TRUCK(POINT1))=LNGTH 00045120
    GOTO 11 00045130
  ENDIF 00045140
  NEXT=FSUCC(NODE) 00045150
  GOTO 10 00045160
11 ISTRT=DEPOT(TRUCK(POINT2)) 00045170
NEXT=FSUCC(ISTRT) 00045180
12 NODE=NEXT 00045190
  IF(NODE.GT.NCITY) THEN 00045200
    IEND=FPRED(NODE) 00045210
    CALL TSP(ISTRT,IEND,FPRED,FSUCC,LNGTH,DIST,ALLOW) 00045220
    IF(LNGTH.GT.DISTLM) GOTO 15 00045230
    FEASLN(TRUCK(POINT2))=LNGTH 00045240
    GOTO 13 00045250
  ENDIF 00045260
  NEXT=FSUCC(NODE) 00045270
  GOTO 12 00045280
C          00045290
C          00045300
C          00045310
C          DETERMINE WHETHER DISTANCE INCREASE IS WITHIN ACCEPTABLE LIMITS. 00045320
C          00045330
C          00045340
13 LTDIST=0 00045350
DO 14 I=1,IROUTE 00045360
14 LTDIST=LTDIST+FEASLN(I) 00045370
  IF(LTDIST.LE.DLIMIT) GOTO 17 00045380
C  ELSE 00045390
15 DO 16 I=NCITY+1,NCITY+IROUTE 00045400
  FSUCC(FPRED(I))=SUCC(FPRED(I)) 00045410
16  FPRED(I)=PRED(I) 00045420
  DO 160 I=1,2 00045430
    IF(I.EQ.1) NODE=DEPOT(TRUCK(POINT1)) 00045440

```

```

        IF(I.EQ.2) NODE=DEPOT(TRUCK(POINT2))          00045450
        FEASLD(TRUCK(NODE))=LOAD(TRUCK(NODE))       00045460
        FEASLN(TRUCK(NODE))=LENGTH(TRUCK(NODE))     00045470
        FSUCC(NODE)=SUCC(NODE)                      00045480
        NEXT=SUCC(NODE)                            00045490
116      NODE=NEXT                                00045500
        IF(NODE.GT.NCITY) GOTO 160                  00045510
        FPRED(NODE)=PRED(NODE)                     00045520
        FSUCC(NODE)=SUCC(NODE)                   00045530
        NEXT=SUCC(NODE)                            00045540
        GOTO 116                                00045550
160      CONTINUE                                00045560
        GOTO 7                                    00045570
C
C
C
C      DETERMINE WHETHER ROUTE LENGTH DEVIATION LIMITS HAVE BEEN    00045580
C      MAINTAINED.                                         00045590
C
C
C      17 FMAXLN=-99          00045600
        FMINLN=999999          00045610
        DO 18 I=1,IROUTE          00045620
          IF(LOKK(DEPOT(I)).NE.0) GOTO 18          00045630
          IF(FEASLN(I).GT.FMAXLN) FMAXLN=FEASLN(I) 00045640
          IF(FEASLN(I).LT.FMINLN.AND.FEASLN(I).NE.0) FMINLN=FEASLN(I)
18      CONTINUE                                00045650
        IF(FMAXLN-FMINLN.GT.LNDVLM) GOTO 15        00045660
C      ELSE                                     00045670
C
C
C
C      CHANGE ROUTE STRUCTURE.                         00045680
C
C
C      MAXLN=FMAXLN          00045690
        MINLN=FMINLN          00045700
        MAXLD=FMAXLD          00045710
        MINLD=FMINLD          00045720
        LNDEV=MAXLN-MINLN      00045730
        LDDEV=MAXLD-MINLD      00045740
        DO 19 I=1,NCITY+IROUTE          00045750
          PRED(I)=FPRED(I)          00045760
        19      SUCC(I)=FSUCC(I)          00045770
        NEXT=NCITY+1          00045780
        INSIDE=0          00045790
20      NODE=NEXT                                00045800
        IF(NODE.EQ.NCITY+1.AND.INSIDE.EQ.1) GOTO 21 00045810
        INSIDE=1          00045820
        IF(NODE.GT.NCITY) THEN          00045830
          ITRK=TRUCK(NODE)          00045840
          ILD=0          00045850
          ILN=0          00045860
        ENDIF          00045870
        ILD=ILD+DEMAND(NODE)          00045880
        IF(NODE.LE.NCITY) ILN=ILN+DIST(NODE,PRED(NODE))+ALLOW 00045890
        CUMLD(NODE)=ILD          00045900
        CUMLN(NODE)=ILN          00045910
        TRUCK(NODE)=ITRK          00045920
        FEASLD(ITRK)=ILD          00045930
        NEXT=SUCC(NODE)          00045940
        IF(NEXT.GT.NCITY) FEASLN(ITRK)=ILN+DIST(NODE,NEXT) 00045950
        GOTO 20          00045960
21      TDIST=0          00045970
        TLOAD=0          00045980
        DO 22 I=1,IROUTE          00045990
          LOAD(I)=FEASLD(I)          00046000
          LENGTH(I)=FEASLN(I)       00046010
          TLOAD=TLOAD+LOAD(I)       00046020
22      TDIST=TDIST+LENGTH(I)          00046030
        NEXT=NCITY+1          00046040

```

```

C                                         00046160
C                                         00046170
C                                         00046180
C   ROTATE                               00046190
C                                         00046200
C                                         00046210
C                                         00046220
C                                         00046230
C                                         00046240
C                                         00046250
C                                         00046260
C                                         00046270
C                                         00046280
C                                         00046290
C                                         00046300
C                                         00046310
C                                         00046320
C                                         00046330
C*****                                         00046340
C                                         00046350
C                                         00046360
C                                         00046370
C                                         00046380
C THIS SUBROUTINE MAKES 'POTENTIAL' TWO-ARC EXCHANGES TO A ROUTE 00046390
C STRUCTURE. NO 'ACTUAL' EXCHANGES ARE MADE.                      00046400
C                                         00046410
C*****                                         00046420
C                                         00046430
C                                         00046440
C CHARACTER*44 PNAME                           00046450
C DOUBLE PRECISION DSEED                      00046460
C INTEGER P1,P2,P3,FIRST,TAIL(120),TALE(20),HEAD(20),GAP1,GAP2, 00046470
*      TYPE,FWD,BACK,START,END,WTLIM,DISTLM,ALLOW,DLIMIT,          00046480
*      D1,D2,D3,D4,D5,D6,DEPOT(20),PRED(120),SUCC(120),          00046490
*      TRUCK(120),DEMAND(0:120),LENGTH(120),LOAD(120),           00046500
*      CUMLD(120),CUMLN(120),TEMTRK(120),FEASLD(20),FEASLN(20), 00046510
*      PERMPR(120),PERMSU(120),PERMTR(120),ISORT(6000),          00046520
*      JSORT(6000),DIST(0:120,0:120),XCOORD(0:120),             00046530
*      YCOORD(0:120),LOKK(120),PDUM(120),SDUM(120)              00046540
C DIMENSION SORT(6000),TRADE(7,500)            00046550
C COMMON IROUTE,NCITY,IRUN,START,END,WTLIM,DISTLM,ALLOW,NPERM,    00046560
*C COUNT,LDDVLM,LNDVLM,DLIMIT,MAXLD,MINLD,MAXLN,MINLN,D1,D2,D3, 00046570
*D5,D6,DEPOT,PDUM,SDUM,TRUCK,DEMAND,LENGTH,LOAD,CUMLD,CUMLN,TEMTRK, 00046580
*FEASLD,FEASLN,PERMPR,PERMSU,PERMTR,ISORT,JSORT,SORT,DIST,XCOORD, 00046590
*YCOORD,LOKK,TRADE,NTRADE,DSEED                00046600
C                                         00046610
C                                         00046620
C                                         00046630
C NULL EXCHANGE:                            00046640
C                                         00046650
C NULL=1                                     00046660
C IF(P1.EQ.PRED(P2).OR.P1.EQ.SUCC(P2)) RETURN 00046670
C IF(P1.GT.NCITY.AND.SUCC(P2).GT.NCITY) RETURN 00046680
C IF(P2.GT.NCITY.AND.SUCC(P1).GT.NCITY) RETURN 00046690
C NULL=0                                     00046700
C                                         00046710
C                                         00046720
C                                         00046730
C REMOVE THE ROUTES INVOLVED IN THE EXCHANGE FROM THE NETWORK AND 00046740
C FORM A SEPARATE NETWORK WITHIN WHICH THE 2-ARC EXCHANGE WILL TAKE 00046750
C PLACE.                                      00046760
C                                         00046770
C                                         00046780
C IS1=SUCC(P1)                                00046790
C IS2=SUCC(P2)                                00046800
C DD 1 I=NCITY+1,NCITY+IRROUTE                 00046810
1 TAIL(DEPOT(TRUCK(PRED(I))))=PRED(I)        00046820
NEXT=NCITY+1                                    00046830
NUM=0                                         00046840
INSIDE=0                                       00046850
3 NODE=NEXT                                     00046860

```

```

IF(NODE.EQ.NCITY+1.AND.INSIDE.EQ.1) GOTO 4          00046870
INSIDE=1                                            00046880
IF(TRUCK(NODE).EQ.TRUCK(P1).OR.TRUCK(NODE).EQ.TRUCK(P2)) THEN 00046890
    NEXT=SUCC(TAIL(NODE))
    GOTO 3                                            00046900
ENDIF                                              00046910
NUM=NUM+1                                           00046920
HEAD(NUM)=NODE                                     00046930
TALE(NUM)=TAIL(NODE)                             00046940
NEXT=SUCC(TAIL(NODE))                           00046950
GOTO 3                                            00046960
00046970
C
4 IF(TRUCK(P1).EQ.TRUCK(P2)) THEN                00046980
    SUCC(TAIL(DEPOT(TRUCK(P1))))=DEPOT(TRUCK(P1)) 00046990
    PRED(DEPOT(TRUCK(P1)))=TAIL(DEPOT(TRUCK(P1))) 00047000
    GOTO 5                                            00047010
ENDIF                                              00047020
    SUCC(TAIL(DEPOT(TRUCK(P1))))=DEPOT(TRUCK(P2)) 00047030
    PRED(DEPOT(TRUCK(P2)))=TAIL(DEPOT(TRUCK(P1))) 00047040
    SUCC(TAIL(DEPOT(TRUCK(P2))))=DEPOT(TRUCK(P1)) 00047050
    PRED(DEPOT(TRUCK(P1)))=TAIL(DEPOT(TRUCK(P2))) 00047060
00047070
C
5 IF(NUM.EQ.1) THEN                            00047080
    SUCC(TALE(1))=HEAD(1)                         00047090
    PRED(HEAD(1))=TALE(1)                         00047100
    GAP1=TALE(1)                                 00047110
    GAP2=HEAD(1)                                 00047120
ENDIF                                              00047130
IF(NUM.GT.1) THEN                            00047140
    DO 6 I=1,NUM-1
        SUCC(TALE(I))=HEAD(I+1)                  00047150
6    PRED(HEAD(I+1))=TALE(I)                   00047160
    SUCC(TALE(NUM))=HEAD(1)                     00047170
    PRED(HEAD(1))=TALE(NUM)                     00047180
    GAP1=TALE(1)                               00047190
    GAP2=SUCC(TALE(1))                         00047200
ENDIF                                              00047210
00047220
C
C
C
C     PERFORM THE EXCHANGE.
C
IS1=SUCC(P1)                                     00047230
IS2=SUCC(P2)                                     00047240
LAST=P1                                         00047250
NEXT=P2                                         00047260
SUCC(P1)=P2                                     00047270
00047280
7 NODE=NEXT                                     00047290
    IP=PRED(NODE)                                00047300
    PRED(NODE)=LAST                            00047310
    IF(NODE.EQ.IS2) GOTO 8                    00047320
    SUCC(NODE)=IP                            00047330
    IF(NODE.EQ.IS1) SUCC(NODE)=IS2           00047340
    LAST=NODE                                 00047350
    NEXT=SUCC(NODE)                           00047360
    GOTO 7                                    00047370
00047380
C
C
C
C     RECONNECT ROUTES INVOLVED IN EXCHANGE BACK INTO ORIGINAL NETWORK.
C
8 PRED(IS2)=LAST                                00047390
    IPRED=PRED(DEPOT(TRUCK(P1)))               00047400
    SUCC(GAP1)=DEPOT(TRUCK(P1))               00047410
    PRED(DEPOT(TRUCK(P1)))=GAP1              00047420
    SUCC(IPRED)=GAP2                          00047430
    PRED(GAP2)=IPRED                         00047440
    RETURN                                     00047450
END                                         00047460
00047470
00047480
00047490
00047500
00047510
00047520
00047530
00047540
00047550
00047560
00047570

```

```

C                                         00047580
C                                         00047590
C*****                                         00047600
C                                         00047610
C     SUBROUTINE LNDV3(LNRLX)             00047620
C                                         00047630
C                                         00047640
C     THIS SUBROUTINE IS USED TO MINIMIZE THE MAXIMUM LENGTH DEVIATION 00047650
C     IN ROUTE LENGTHS VIA THE THREE-ARC BRANCH EXCHANGE METHOD.        00047660
C                                         00047670
C*****                                         00047680
C                                         00047690
C                                         00047700
C     CHARACTER*1 MODE                  00047710
C     CHARACTER*44 PNAME                00047720
C     DOUBLE PRECISION DSEED          00047730
C     INTEGER START,END,WTLIM,DISTLM,ALLOW,DLIMIT,D1,D2,D3,D4,D5,D6,    00047740
*      DEPOT(20),PRED(120),SUCC(120),FPRED(120),FSUCC(120),           00047750
*      TRUCK(120),DEMAND(0:120),LENGTH(120),LOAD(120),CUMLD(120),00047760
*      CUMLN(120),TEMTRK(120),FEASLD(20),FEASLN(20),PERMPR(120), 00047770
*      PERMSU(120),PERMTR(120),ISORT(6000),JSORT(6000),            00047780
*      DIST(0:120,0:120),XCOORD(0:120),YCOORD(0:120),TLOAD,TDIST,00047790
*      P1,P2,P3,FMAXLD,FMINLD,FMAXLN,FMINLN,DEPDT4,TR(3),       00047800
*      TRY(120,120),FLDDEV,TWTLIM           00047810
C     DIMENSION SORT(6000),LOKK(120)        00047820
C     DIMENSION TRADE(7,500)                 00047830
C     COMMON IROUTE,NCITY,IRUN,START,END,WTLIM,DISTLM,ALLOW,NPERM,    00047840
*ICOUNT,LDDVLM,LNDVLM,DLIMIT,MAXLD,MINLD,MAXLN,MINLN,D1,D2,D3,D4, 00047850
*D5,D6,DEPOT,PRED,SUCC,TRUCK,DEMAND,LENGTH,LOAD,CUMLD,CUMLN,TEMTRK,00047860
*FEASLD,FEASLN,PERMPR,PERMSU,PERMTR,ISORT,JSORT,SORT,DIST,XCOORD, 00047870
*YCOORD,LOKK,TRADE,NTRADE,DSEED          00047880
C                                         00047890
C                                         00047900
C                                         00047910
C     NTRADE=0                           00047920
C     MODE='F'                          00047930
C     TLOAD=0                           00047940
C     TDIST=0                           00047950
C     START=GGUBFS(DSEED)*NCITY+1       00047960
C     END=PRED(START)                  00047970
C     TWTLIM=MINO(WTLIM,MAXLD+LDDVLM)  00047980
C     DO 1 I=1,IROUTE                 00047990
C       FEASLD(I)=LOAD(I)              00048000
C       FEASLN(I)=LENGTH(I)            00048010
C       TDIST=TDIST+LENGTH(I)          00048020
1   TLOAD=TLOAD+LOAD(I)               00048030
C     DO 2 I=1,NCITY+IROUTE           00048040
C       FPRED(I)=PRED(I)              00048050
2   FSUCC(I)=SUCC(I)                00048060
C     DO 3 I=1,NCITY+IROUTE           00048070
C       DO 3 J=1,NCITY+IROUTE         00048080
3   TRY(I,J)=0                      00048090
C                                         00048100
C                                         00048110
C                                         00048120
C     DETERMINE LOAD AND LENGTH DEVIATIONS 00048130
C                                         00048140
C                                         00048150
C     LNDEV=MAXLN-MINLN              00048160
C     LDDEV=MAXLD-MINLD              00048170
C                                         00048180
C                                         00048190
C                                         00048200
C     BEGIN ITERATIONS              00048210
C                                         00048220
C                                         00048230
C     P1=PRED(START)                00048240
6  P1=SUCC(P1)                     00048250
IF(P1.EQ.PRED(PRED(END))) THEN      00048260
  RETURN                            00048270
ENDIF                             00048280

```

```

IF(LOKK(P1).EQ.1) GOTO 6          00048290
P2=P1                           00048300
7 P2=SUCC(P2)                   00048310
IF(P2.EQ.PRED(END)) GOTD 6       00048320
IF(LOKK(P2).EQ.1) GOTO 7         00048330
P3=P2                           00048340
8 P3=SUCC(P3)                   00048350
IF(P3.EQ.END) GOTO 7             00048360
IF(LOKK(P3).EQ.1) GOTO 8         00048370
C                               00048380
C                               00048390
C                               00048400
C IF ALL ARCS ARE IN THE SAME ROUTE, IGNORE THE EXCHANGE 00048410
C                               00048420
C                               00048430
C IF(TRUCK(P1).EQ.TRUCK(P2).AND.TRUCK(P1).EQ.TRUCK(P3)) GOTO 8 00048440
C                               00048450
C                               00048460
C                               00048470
C DETERMINE WHETHER AT LEAST ONE OF THE ARCS IS IN THE LONG ROUTE 00048480
C OR IN THE SHORT ROUTE. IF NOT, IGNORE THE EXCHANGE.        00048490
C                               00048500
C                               00048510
C IF(LOAD(TRUCK(P1)).LE.0.OR.LOAD(TRUCK(P2)).LE.0.OR.      00048520
*   LOAD(TRUCK(P3)).LE.0) GOTO 8                         00048530
IF(LENGTH(TRUCK(P1)).EQ.MAXLN.OR.LENGTH(TRUCK(P1)).EQ.MINLN) 00048540
*   GOTO 99                                         00048550
IF(LENGTH(TRUCK(P2)).EQ.MAXLN.OR.LENGTH(TRUCK(P2)).EQ.MINLN) 00048560
*   GOTO 99                                         00048570
IF(LENGTH(TRUCK(P3)).EQ.MAXLN.OR.LENGTH(TRUCK(P3)).EQ.MINLN) 00048580
*   GOTO 99                                         00048590
C ELSE                                           00048600
GOTO 8                                         00048610
C                               00048620
C                               00048630
99 CONTINUE                                     00048640
IF(LNRLX.GT.0) THEN                           00048650
  IF(P1.GT.NCITY) GOTO 8                     00048660
  IF(P2.GT.NCITY) GOTO 8                     00048670
  IF(P3.GT.NCITY) GOTO 8                     00048680
ENDIF                                         00048690
809 FORMAT(1H.,3I7)                           00048700
IF(TRUCK(P1).NE.TRUCK(P2).AND.TRUCK(P1).NE.TRUCK(P3).AND. 00048710
*   TRUCK(P3).NE.TRUCK(P2)) THEN            00048720
  LONER=0                                       00048730
  GOTO 9                                         00048740
ENDIF                                         00048750
IF(TRUCK(P2).EQ.TRUCK(P3)) THEN            00048760
  LONER=1                                       00048770
  GOTO 11                                       00048780
ENDIF                                         00048790
IF(TRUCK(P1).EQ.TRUCK(P3)) THEN            00048800
  LONER=2                                       00048810
  GOTO 11                                       00048820
ENDIF                                         00048830
IF(TRUCK(P1).EQ.TRUCK(P2)) THEN            00048840
  LONER=3                                       00048850
  GOTO 11                                       00048860
ENDIF                                         00048870
C                               00048880
C                               00048890
C                               00048900
C TYPE I EXCHANGE - LOAD FEASIBILITY TEST      00048910
C                               00048920
C                               00048930
9 NTYPE=1                                       00048940
  IF(CUMLD(P1)+CUMLD(P2).GT.TWTL) GOTO 10      00048950
  IF(LOAD(TRUCK(P1))-CUMLD(P1)+CUMLD(P3).GT.TWTL) GOTO 10 00048960
  IF(LOAD(TRUCK(P2))-CUMLD(P2)+LOAD(TRUCK(P3))-CUMLD(P3).GT.TWTL) 00048970
*   GOTO 10                                         00048980
  FEASLD(TRUCK(P1))=CUMLD(P1)+CUMLD(P2)           00048990

```

```

C
C
C      TYPE II EXCHANGE - LOAD FEASIBILITY TEST
C
C
10 NTYPE=2
    IF(CUMLD(P1)+LOAD(TRUCK(P2))-CUMLD(P2).GT.TWTL) GOTO 11
    IF(CUMLD(P2)+CUMLD(P3).GT.TWTL) GOTO 11
    IF(LOAD(TRUCK(P1))-CUMLD(P1)+LOAD(TRUCK(P3))-CUMLD(P3).GT.TWTL) GOTO 11
    *      FEASLD(TRUCK(P1))=CUMLD(P1)+LOAD(TRUCK(P2))-CUMLD(P2)
    *      FEASLD(TRUCK(P2))=CUMLD(P2)+CUMLD(P3)
    *      FEASLD(TRUCK(P3))=LOAD(TRUCK(P1))-CUMLD(P1)+LOAD(TRUCK(P3))
    *                  -CUMLD(P3)
    NUMTRK=3
    TR(1)=P1
    TR(2)=P2
    TR(3)=P3
    GOTO 50

C
C
C      TYPE III EXCHANGE - LOAD FEASIBILITY TEST
C
C
11 NTYPE=3
    IF(LONER.EQ.0) THEN
        IF(P1.GT.NCITY.AND.P2.GT.NCITY.AND.P3.GT.NCITY) GOTO 12
        IF(CUMLD(P1)+LOAD(TRUCK(P2))-CUMLD(P2).GT.TWTL) GOTO 12
        IF(LOAD(TRUCK(P1))-CUMLD(P1)+CUMLD(P3).GT.TWTL) GOTO 12
        IF(CUMLD(P2)+LOAD(TRUCK(P3))-CUMLD(P3).GT.TWTL) GOTO 12
        FEASLD(TRUCK(P1))=CUMLD(P1)+LOAD(TRUCK(P2))-CUMLD(P2)
        FEASLD(TRUCK(P2))=LOAD(TRUCK(P1))-CUMLD(P1)+CUMLD(P3)
        FEASLD(TRUCK(P3))=CUMLD(P2)+LOAD(TRUCK(P3))-CUMLD(P3)
        NUMTRK=3
        TR(1)=P1
        TR(2)=P2
        TR(3)=P3
        GOTO 50
    ENDIF
    IF(LONER.EQ.1) THEN
        IF(LOAD(TRUCK(P1))+CUMLD(P3)-CUMLD(P2).GT.TWTL) GOTO 13
        FEASLD(TRUCK(P1))=LOAD(TRUCK(P1))+CUMLD(P3)-CUMLD(P2)
        FEASLD(TRUCK(P2))=LOAD(TRUCK(P2))-CUMLD(P3)+CUMLD(P2)
        NUMTRK=2
        TR(1)=P1
        TR(2)=P2
        GOTO 50
    ENDIF
    IF(LONER.EQ.3) THEN
        IF(LOAD(TRUCK(P3))+CUMLD(P2)-CUMLD(P1).GT.TWTL) GOTO 14
        FEASLD(TRUCK(P2))=LOAD(TRUCK(P2))-CUMLD(P2)+CUMLD(P1)
        FEASLD(TRUCK(P3))=LOAD(TRUCK(P3))+CUMLD(P2)-CUMLD(P1)
        NUMTRK=2
        TR(1)=P2
        TR(2)=P3
        GOTO 50
    ENDIF
    IF(LONER.EQ.2) THEN
        IF(LOAD(TRUCK(P2))+CUMLD(P1)-CUMLD(P3).GT.TWTL) GOTO 13
        FEASLD(TRUCK(P2))=LOAD(TRUCK(P2))+CUMLD(P1)-CUMLD(P3)
        FEASLD(TRUCK(P1))=LOAD(TRUCK(P1))-CUMLD(P1)+CUMLD(P3)
        NUMTRK=2

```

```

      TR(1)=P2          00049710
      TR(2)=P1          00049720
      GOTO 50           00049730
      ENDIF             00049740
C
C
C   TYPE IV EXCHANGE - LOAD FEASIBILITY TEST
C
C
12 NTYPE=4          00049750
      IF(CUMLD(P1)+CUMLD(P3).GT.TWTL) GOTO 13 00049760
      IF(LOAD(TRUCK(P1))-CUMLD(P1)+LOAD(TRUCK(P2))-CUMLD(P2).GT.TWTL) 00049830
      * GOTO 13          00049840
      IF(CUMLD(P2)+LOAD(TRUCK(P3))-CUMLD(P3).GT.TWTL) GOTO 13 00049850
      FEASLD(TRUCK(P1))=CUMLD(P1)+CUMLD(P3) 00049860
      FEASLD(TRUCK(P2))=LOAD(TRUCK(P1))-CUMLD(P1)+LOAD(TRUCK(P2)) 00049870
      * -CUMLD(P2)          00049880
      FEASLD(TRUCK(P3))=CUMLD(P2)+LOAD(TRUCK(P3))-CUMLD(P3) 00049890
      NUMTRK=3           00049900
      TR(1)=P1           00049910
      TR(2)=P2           00049920
      TR(3)=P3           00049930
      GOTO 50             00049940
C
C
C   TYPE V EXCHANGE - LOAD FEASIBILITY TEST
C
C
13 NTYPE=5          00049950
      IF(TRY(P1,P2).EQ.1) GOTO 14             00049960
      TRY(P1,P2)=1          00049970
      IF(LONER.EQ.3) GOTO 14             00049980
      IF(CUMLD(P1)+CUMLD(P2).GT.TWTL) GOTO 14 00049990
      IF(LOAD(TRUCK(P1))-CUMLD(P1)+LOAD(TRUCK(P2))-CUMLD(P2).GT.TWTL) 00050000
      * GOTO 14             00050010
      FEASLD(TRUCK(P1))=CUMLD(P1)+CUMLD(P2) 00050020
      FEASLD(TRUCK(P2))=LOAD(TRUCK(P1))-CUMLD(P1)+LOAD(TRUCK(P2)) 00050030
      * -CUMLD(P2)          00050040
      FEASLD(TRUCK(P3))=CUMLD(P2)+LOAD(TRUCK(P3))-CUMLD(P3) 00050050
      NUMTRK=2           00050060
      TR(1)=P1           00050070
      TR(2)=P2           00050080
      GOTO 50             00050090
C
C
C   TYPE VI EXCHANGE - LOAD FEASIBILITY TEST
C
C
14 NTYPE=6          00050100
      IF(TRY(P2,P3).EQ.1) GOTO 15             00050110
      TRY(P2,P3)=1          00050120
      IF(LONER.EQ.1) GOTO 15             00050130
      IF(CUMLD(P2)+CUMLD(P3).GT.TWTL) GOTO 15 00050140
      IF(LOAD(TRUCK(P2))-CUMLD(P2)+LOAD(TRUCK(P3))-CUMLD(P3).GT.TWTL) 00050150
      * GOTO 15             00050160
      FEASLD(TRUCK(P2))=CUMLD(P2)+CUMLD(P3) 00050170
      FEASLD(TRUCK(P3))=LOAD(TRUCK(P2))-CUMLD(P2)+LOAD(TRUCK(P3)) 00050180
      * -CUMLD(P3)          00050190
      FEASLD(TRUCK(P4))=CUMLD(P3)+LOAD(TRUCK(P4))-CUMLD(P4) 00050200
      NUMTRK=2           00050210
      TR(1)=P2           00050220
      TR(2)=P3           00050230
      GOTO 50             00050240
C
C
C   TYPE VII EXCHANGE - LOAD FEASIBILITY TEST
C
C
15 NTYPE=7          00050250
      IF(TRY(P3,P4).EQ.1) GOTO 16             00050260
      TRY(P3,P4)=1          00050270
      IF(LONER.EQ.1) GOTO 16             00050280
      IF(CUMLD(P3)+CUMLD(P4).GT.TWTL) GOTO 16 00050290
      IF(LOAD(TRUCK(P3))-CUMLD(P3)+LOAD(TRUCK(P4))-CUMLD(P4).GT.TWTL) 00050300
      * GOTO 16             00050310
      FEASLD(TRUCK(P3))=CUMLD(P3)+CUMLD(P4) 00050320
      FEASLD(TRUCK(P4))=LOAD(TRUCK(P3))-CUMLD(P3)+LOAD(TRUCK(P4)) 00050330
      * -CUMLD(P4)          00050340
      FEASLD(TRUCK(P5))=CUMLD(P4)+LOAD(TRUCK(P5))-CUMLD(P5) 00050350
      FEASLD(TRUCK(P6))=CUMLD(P5)+LOAD(TRUCK(P6))-CUMLD(P6) 00050360
      * -CUMLD(P5)          00050370
      FEASLD(TRUCK(P7))=CUMLD(P6)+LOAD(TRUCK(P7))-CUMLD(P7) 00050380
      FEASLD(TRUCK(P8))=CUMLD(P7)+LOAD(TRUCK(P8))-CUMLD(P8) 00050390
      * -CUMLD(P7)          00050400
      FEASLD(TRUCK(P9))=CUMLD(P8)+LOAD(TRUCK(P9))-CUMLD(P9) 00050410
      NUMTRK=2           00050420
      TR(1)=P3           00050430
      TR(2)=P4           00050440
      GOTO 50             00050450

```

```

IF(TRY(P1,P3).EQ.1) GOTO 8          00050420
TRY(P1,P3)=1                      00050430
IF(LONER.EQ.2) GOTO 8              00050440
IF(CUMLD(P1)+CUMLD(P3).GT.TWTL) GOTO 8 00050450
IF(LOAD(TRUCK(P1))-CUMLD(P1)+LOAD(TRUCK(P3))-CUMLD(P3).GT.TWTL) 00050460
*   GOTO 8
FEASLD(TRUCK(P1))=CUMLD(P1)+CUMLD(P3) 00050480
FEASLD(TRUCK(P3))=LOAD(TRUCK(P1))-CUMLD(P1)+LOAD(TRUCK(P3)) 00050490
*           -CUMLD(P3)               00050500
NUMTRK=2                           00050510
TR(1)=P1                          00050520
TR(2)=P3                          00050530
GOTO 50                           00050540
C
C
C   DETERMINE WHETHER LOAD DEVIATION LIMITS HAVE BEEN MAINTAINED 00050550
C
C
C   50 FMAXLD=-99                 00050560
FMINLD=999999                     00050570
DO 51 I=1,IROUTE                  00050580
  IF(LOKK(DEPOT(I)).NE.0) GOTO 51 00050590
  IF(FEASLD(I).GT.FMAXLD) FMAXLD=FEASLD(I) 00050600
  IF(FEASLD(I).LT.FMINLD.AND.FEASLD(I).NE.0) FMINLD=FEASLD(I) 00050610
51 CONTINUE                         00050620
  IF(LNRLX.GT.0.AND.FMAXLD-FMINLD.GT.LDDVLM) GOTO 1111 00050630
C
C
C   ESTABLISH POTENTIAL ROUTE STRUCTURE RESULTING FROM EXCHANGE 00050640
C
C
C   IF(NTYPE.LE.4) CALL FXCH3(P1,P2,P3,FPRED,FSUCC,NTYPE) 00050650
IF(NTYPE.EQ.5) CALL FXCH2(P1,P2,FPRED,FSUCC,NULL) 00050660
IF(NTYPE.EQ.6) CALL FXCH2(P2,P3,FPRED,FSUCC,NULL) 00050670
IF(NTYPE.EQ.7) CALL FXCH2(P1,P3,FPRED,FSUCC,NULL) 00050680
IF(NTYPE.GT.4.AND.NULL.EQ.1) THEN 00050690
1111  FEASLD(TRUCK(P1))=LOAD(TRUCK(P1)) 00050700
    FEASLD(TRUCK(P2))=LOAD(TRUCK(P2)) 00050710
    FEASLD(TRUCK(P3))=LOAD(TRUCK(P3)) 00050720
    IF (LONER.EQ.0) GOTO (10,11,12,13,14,15,8), NTYPE 00050730
    IF(LONER.EQ.1) GOTO (11,11,13,13,15,15,8), NTYPE 00050740
    IF(LONER.EQ.2) GOTO (11,11,13,13,14,8,8), NTYPE 00050750
    IF(LONER.EQ.3) GOTO (11,11,14,14,14,15,8),NTYPE 00050760
  ENDIF
C
C
C   SOLVE TSP FOR EACH ROUTE AFFECTED BY THE EXCHANGE 00050770
C
C
C   DO 55 I=1,NUMTRK                00050780
    ISTRT=DEPOT(TRUCK(TR(I))) 00050790
    NEXT=FSUCC(ISTRT)          00050800
52  NODE=NEXT                      00050810
    IF(NODE.GT.NCITY) THEN      00050820
      IEND=FPRED(NODE)        00050830
      CALL TSP(ISTRT,IEND,FPRED,FSUCC,LNGTH,DIST,ALLOW) 00050840
      IF(LNGTH.GT.DISTLM) GOTO 58 00050850
      FEASLN(TRUCK(TR(I)))=LNGTH 00050860
      GOTO 55                  00050870
    ENDIF
    NEXT=FSUCC(NODE)          00050880
  GOTO 52                          00050890
55 CONTINUE                         00050900
C
C
C   DETERMINE DISTANCE INCREASE. 00050910
C
C

```

```

C
      LTDIST=0          00051130
      DO 56 I=1,IROUTE 00051140
      56 LTDIST=LTDIST+FEASLN(I) 00051150
C
C
C      DETERMINE WHETHER ROUTE LENGTH DEVIATION IS REDUCED 00051160
C
C
      FMAXLN=-99        00051170
      FMINLN=999999      00051180
      DO 57 I=1,IROUTE 00051190
          IF(LOKK(DEPOT(I)).NE.0) GOTO 57 00051200
          IF(FEASLN(I).GT.FMAXLN) FMAXLN=FEASLN(I) 00051210
          IF(FEASLN(I).LT.FMINLN.AND.FEASLN(I).NE.0) FMINLN=FEASLN(I) 00051220
      57 CONTINUE        00051230
          IF(FMAXLN-FMINLN.GE.LNDEV+LNRLX) THEN 00051240
      58   DO 59 I=NCITY+1,NCITY+IROUTE 00051250
          FSUCC(FPRED(I))=SUCC(FPRED(I)) 00051260
      59   FPRED(I)=PRED(I) 00051270
          DO 61 I=1,NUMTRK 00051280
              FEASLD(TRUCK(TR(I)))=LOAD(TRUCK(TR(I))) 00051290
              FEASLN(TRUCK(TR(I)))=LENGTH(TRUCK(TR(I))) 00051300
              NODE=DEPOT(TRUCK(TR(I))) 00051310
              FPRED(NODE)=PRED(NODE) 00051320
              FSUCC(NODE)=SUCC(NODE) 00051330
              NEXT=SUCC(NODE) 00051340
      60   NODE=NEXT 00051350
          IF(NODE.GT.NCITY) GOTO 61 00051360
          FPRED(NODE)=PRED(NODE) 00051370
          FSUCC(NODE)=SUCC(NODE) 00051380
          NEXT=SUCC(NODE) 00051390
          GOTO 60 00051400
      61   CONTINUE        00051410
          IF(LONER.EQ.0) GOTO (10,11,12,13,14,15,8), NTYPE 00051420
          IF(LONER.EQ.1) GOTO (11,11,13,13,15,15,8), NTYPE 00051430
          IF(LONER.EQ.2) GOTO (11,11,13,13,14,8,8), NTYPE 00051440
          IF(LONER.EQ.3) GOTO (11,11,14,14,14,15,8), NTYPE 00051450
      ENDIF 00051460
C
C
C      TRADEOFF ANALYSIS 00051470
C
C
      IF(LNRLX.GT.0) GOTO 62 00051480
      IF(LTDIST.GT.DLIMIT.OR.FMAXLD-FMINLD.GT.LDDVLM) THEN 00051490
          NTRADE=NTRADE+1 00051500
          IF(NTRADE.GT.500) NTRADE=500 00051510
          TRADE(1,NTRADE)=LNDEV-(FMAXLN-FMINLN) 00051520
          TRADE(1,NTRADE)=-TRADE(1,NTRADE) 00051530
          TRADE(2,NTRADE)=LTDIST-TDIST 00051540
          TRADE(3,NTRADE)=FMAXLD-FMINLD-LDDEV 00051550
          TRADE(4,NTRADE)=P1 00051560
          TRADE(5,NTRADE)=P2 00051570
          TRADE(6,NTRADE)=P3 00051580
          TRADE(7,NTRADE)=NTYPE 00051590
      ENDIF 00051600
      62 IF(LTDIST.GT.DLIMIT) GOTO 58 00051610
      IF(FMAXLD-FMINLD.GT.LDDVLM) GOTO 58 00051620
C     ELSE 00051630
C
C
C      CHANGE ROUTE STRUCTURE 00051640
C
C
      MAXLN=FMAXLN 00051650
      MINLN=FMINLN 00051660
      MAXLD=FMAXLD 00051670
      00051680
      00051690
      00051700
      00051710
      00051720
      00051730
      00051740
      00051750
      00051760
      00051770
      00051780
      00051790
      00051800
      00051810
      00051820
      00051830

```

```

MINLD=FMINLD          00051840
LNDEV=MAXLN-MINLN    00051850
LDDEV=MAXLD-MINLD    00051860
DO 66 I=1,NCITY+IROUTE 00051870
    PRED(I)=FPRED(I)   00051880
66    SUCC(I)=FSUCC(I) 00051890
    NEXT=NCITY+1        00051900
    INSIDE=0            00051910
67 NODE=NEXT          00051920
    IF(NODE.EQ.NCITY+1.AND.INSIDE.EQ.1) GOTO 68 00051930
    INSIDE=1            00051940
    IF(NODE.GT.NCITY) THEN 00051950
        ITRK=TRUCK(NODE)
        ILD=0              00051960
        ILN=0              00051970
        ENDIF              00051980
        ILD=ILD+DEMAND(NODE) 00051990
        IF(NODE.LE.NCITY) ILN=ILN+DIST(NODE,PRED(NODE))+ALLOW 00052000
        CUMLD(NODE)=ILD    00052010
        CUMLN(NODE)=ILN    00052020
        TRUCK(NODE)=ITRK   00052030
        FEASLD(ITRK)=ILD   00052040
        NEXT=SUCC(NODE)    00052050
        IF(NEXT.GT.NCITY) FEASLN(ITRK)=ILN+DIST(NODE,NEXT) 00052060
        GOTO 67            00052070
68 TDIST=0             00052080
    TLOAD=0              00052090
    DO 69 I=1,IROUTE    00052100
        LOAD(I)=FEASLD(I) 00052110
        LENGTH(I)=FEASLN(I) 00052120
        TLOAD=TLOAD+LOAD(I) 00052130
69    TDIST=TDIST+LENGTH(I) 00052140
C
C
C
C     ROTATE             00052150
C
C
TWLIM=MINO(WTLIM,MAXLD+LDDVLM) 00052220
IRLX=LNRLX            00052230
LNRLX=0              00052240
NTRADE=0              00052250
ISTART=END            00052260
IF(IRLX.GT.0) ISTART=DEPOT(TRUCK(PRED(ISTART))) 00052270
END=PRED(ISTART)       00052280
START=ISTART          00052290
P1=END                00052300
DO 70 I=1,NCITY+IROUTE 00052310
    DO 70 J=1,NCITY+IROUTE 00052320
70 TRY(I,J)=0          00052330
    GOTO 6              00052340
    END                 00052350
C
C
C
C*****SUBROUTINE LDDV3(LDRLX)          00052360
C
C
C     THIS SUBROUTINE IS USED TO MINIMIZE THE MAXIMUM LOAD DEVIATION 00052440
C     IN ROUTE LOADS VIA THE THREE-ARC BRANCH EXCHANGE METHOD.        00052450
C
C*****CHARACTER*1 MODE                  00052460
CHARACTER*44 PNAME            00052470
DOUBLE PRECISION DSEED         00052480
INTEGER START,END,WTLIM,DISTLM,ALLOW,DLIMIT,D1,D2,D3,D4,D5,D6, 00052490
*           DEPOT(20),PRED(120),SUCC(120),FPRED(120),FSUCC(120), 00052500
*                                         00052510
*                                         00052520
*                                         00052530
*                                         00052540

```

```

*      TRUCK(120),DEMAND(0:120),LENGTH(120),LOAD(120),CUMLD(120),00052550
*      CUMLN(120),TEMTRK(120),FEASLD(20),FEASLN(20),PERMPR(120), 00052560
*      PERMSU(120),PERMTR(120),ISORT(6000),JSORT(6000),          00052570
*      DIST(O:120,O:120),XCOORD(O:120),YCOORD(O:120),TLOAD,TDIST,00052580
*      P1,P2,P3,FMAXLD,FMINLD,FMAXLN,FMINLN,DEPOT4,TR(3),       00052590
*      TRY(120,120),TWLTM                                         00052600
DIMENSION SORT(6000),LOKK(120)                                     00052610
DIMENSION TRADE(7,500)                                           00052620
COMMON IROUTE,NCITY,IRUN,START,END,WTLIM,DISTLM,ALLOW,NPERM,    00052630
*ICOUNT,LDDVLM,LNDVLM,DLIMIT,MAXLD,MINLD,MAXLN,MINLN,D1,D2,D3,D4, 00052640
*D5,D6,DEPOT,PRED,SUCC,TRUCK,DEMAND,LENGTH,LOAD,CUMLD,CUMLN,TEMTRK,00052650
*FEASLD,FEASLN,PERMPR,PERMSU,PERMTR,ISORT,JSORT,SORT,DIST,XCOORD, 00052660
*YCOORD,LOKK,TRADE,NTRADE,DSEED                                00052670
C
C
C
C      NTRADE=0                                              00052720
C      MODE='F'                                              00052730
C      TLOAD=0                                              00052740
C      TDIST=0                                              00052750
C      START=GGUBFS(DSEED)*NCITY+1                           00052760
C      END=PRED(START)                                       00052770
C      DO 1 I=1,IRROUTE                                     00052780
C          FEASLD(I)=LOAD(I)                                 00052790
C          FEASLN(I)=LENGTH(I)                             00052800
C          TDIST=TDIST+LENGTH(I)                           00052810
1     TLOAD=TLOAD+LOAD(I)                                 00052820
C      DO 2 I=1,NCITY+IRROUTE                           00052830
C          FPRED(I)=PRED(I)                               00052840
2     FSUCC(I)=SUCC(I)                                 00052850
C      DO 3 I=1,NCITY+IRROUTE                           00052860
C          DO 3 J=1,NCITY+IRROUTE                           00052870
3     TRY(I,J)=0                                      00052880
C
C
C
C      DETERMINE LOAD AND LENGTH DEVIATIONS            00052910
C
C
C      LNDEV=MAXLN-MINLN                                00052950
C      LDDEV=MAXLD-MINLD                                00052960
C
C
C      BEGIN ITERATIONS                                00053000
C
C
C      TWLTM=MINO(WTLIM,MAXLD+LDDEV+LDRLX)           00053030
C      P1=PRED(START)                                    00053040
6     P1=SUCC(P1)                                      00053050
C      IF(P1.EQ.PRED(PRED(END))) THEN                 00053060
C          RETURN                                         00053070
C          ENDIF                                         00053080
C          IF(LOKK(P1).EQ.1) GOTO 6                     00053090
C          P2=P1                                         00053100
7     P2=SUCC(P2)                                      00053110
C          IF(P2.EQ.PRED(END)) GOTO 6                  00053120
C          IF(LOKK(P2).EQ.1) GOTO 7                     00053130
C          P3=P2                                         00053140
8     P3=SUCC(P3)                                      00053150
C          IF(P3.EQ.END) GOTO 7                        00053160
C          IF(LOKK(P3).EQ.1) GOTO 8                     00053170
C
C
C      IF ALL ARCS ARE IN THE SAME ROUTE, IGNORE THE EXCHANGE 00053210
C
C
C      IF(TRUCK(P1).EQ.TRUCK(P2).AND.TRUCK(P1).EQ.TRUCK(P3)) GOTO 8 00053240
C
C
C

```

```

C                                         00053260
C                                         00053270
C DETERMINE WHETHER AT LEAST ONE OF THE ARCS IS IN THE HEAVY ROUTE 00053280
C OR IN THE LIGHT ROUTE. IF NOT, IGNORE THE EXCHANGE. 00053290
C                                         00053300
C                                         00053310
C
C     IF(LOAD(TRUCK(P1)).LE.0.OR.LOAD(TRUCK(P2)).LE.0.OR. 00053320
*     LOAD(TRUCK(P3)).LE.0) GOTO 8 00053330
IF(LOAD(TRUCK(P1)).EQ.MAXLD.OR.LOAD(TRUCK(P1)).EQ.MINLD) 00053340
*     GOTO 99 00053350
IF(LOAD(TRUCK(P2)).EQ.MAXLD.OR.LOAD(TRUCK(P2)).EQ.MINLD) 00053360
*     GOTO 99 00053370
IF(LOAD(TRUCK(P3)).EQ.MAXLD.OR.LOAD(TRUCK(P3)).EQ.MINLD) 00053380
*     GOTO 99 00053390
C     ELSE 00053400
C     GOTO 8 00053410
C
C
C 99 CONTINUE 00053420
IF(LDRXLX.GT.0) THEN 00053430
    IF(P1.GT.NCITY) GOTO 8 00053440
    IF(P2.GT.NCITY) GOTO 8 00053450
    IF(P3.GT.NCITY) GOTO 8 00053460
ENDIF 00053470
IF(TRUCK(P1).NE.TRUCK(P2).AND.TRUCK(P1).NE.TRUCK(P3).AND. 00053480
*     TRUCK(P3).NE.TRUCK(P2)) THEN 00053490
    LONER=0 00053500
    GOTO 9 00053510
ENDIF 00053520
IF(TRUCK(P2).EQ.TRUCK(P3)) THEN 00053530
    LONER=1 00053540
    GOTO 11 00053550
ENDIF 00053560
IF(TRUCK(P1).EQ.TRUCK(P3)) THEN 00053570
    LONER=2 00053580
    GOTO 11 00053590
ENDIF 00053600
IF(TRUCK(P1).EQ.TRUCK(P2)) THEN 00053610
    LONER=3 00053620
    GOTO 11 00053630
ENDIF 00053640
IF(TRUCK(P1).EQ.TRUCK(P3)) THEN 00053650
    LONER=3 00053660
    GOTO 11 00053670
ENDIF 00053680
C
C
C
C     TYPE I EXCHANGE - LOAD FEASIBILITY TEST 00053690
C                                         00053700
C                                         00053710
C                                         00053720
C
C 9 NTYPE=1 00053730
    IF(CUMLD(P1)+CUMLD(P2).GT.TWTLIM) GOTO 10 00053740
    IF(LOAD(TRUCK(P1))-CUMLD(P1)+CUMLD(P3).GT.TWTLIM) GOTO 10 00053750
    IF(LOAD(TRUCK(P2))-CUMLD(P2)+LOAD(TRUCK(P3))-CUMLD(P3).GT.TWTLIM) 00053760
*     GOTO 10 00053770
    FEASLD(TRUCK(P1))=CUMLD(P1)+CUMLD(P2) 00053780
    FEASLD(TRUCK(P2))=LOAD(TRUCK(P1))-CUMLD(P1)+CUMLD(P3) 00053790
    FEASLD(TRUCK(P3))=LOAD(TRUCK(P2))-CUMLD(P2)+LOAD(TRUCK(P3)) 00053800
*     -CUMLD(P3) 00053810
    NUMTRK=3 00053820
    TR(1)=P1 00053830
    TR(2)=P2 00053840
    TR(3)=P3 00053850
    GOTO 50 00053860
C
C
C
C     TYPE II EXCHANGE - LOAD FEASIBILITY TEST 00053870
C                                         00053880
C                                         00053890
C
C 10 NTYPE=2 00053900
    IF(CUMLD(P1)+LOAD(TRUCK(P2))-CUMLD(P2).GT.TWTLIM) GOTO 11 00053910
    IF(CUMLD(P2)+CUMLD(P3).GT.TWTLIM) GOTO 11 00053920
    IF(LOAD(TRUCK(P1))-CUMLD(P1)+LOAD(TRUCK(P3))-CUMLD(P3).GT.TWTLIM) 00053930

```

```

*      GOTO 11                                         00053970
FEASLD(TRUCK(P1))=CUMLD(P1)+LOAD(TRUCK(P2))-CUMLD(P2) 00053980
FEASLD(TRUCK(P2))=CUMLD(P2)+CUMLD(P3)                00053990
*      FEASLD(TRUCK(P3))=LOAD(TRUCK(P1))-CUMLD(P1)+LOAD(TRUCK(P3)) 00054000
*          -CUMLD(P3)                                00054010
NUMTRK=3                                              00054020
TR(1)=P1                                             00054030
TR(2)=P2                                             00054040
TR(3)=P3                                             00054050
GOTO 50                                              00054060
C
C
C      TYPE III EXCHANGE - LOAD FEASIBILITY TEST      00054070
C
C
11 NTYPE=3                                         00054080
IF(LONER.EQ.0) THEN                               00054090
    IF(P1.GT.NCITY.AND.P2.GT.NCITY.AND.P3.GT.NCITY) GOTO 12 00054100
    IF(CUMLD(P1)+LOAD(TRUCK(P2))-CUMLD(P2).GT.TWTLIM) GOTO 12 00054110
    IF(LOAD(TRUCK(P1))-CUMLD(P1)+CUMLD(P3).GT.TWTLIM) GOTO 12 00054120
    IF(CUMLD(P2)+LOAD(TRUCK(P3))-CUMLD(P3).GT.TWTLIM) GOTO 12 00054130
    FEASLD(TRUCK(P1))=CUMLD(P1)+LOAD(TRUCK(P2))-CUMLD(P2) 00054140
    FEASLD(TRUCK(P2))=LOAD(TRUCK(P1))-CUMLD(P1)+CUMLD(P3) 00054150
    FEASLD(TRUCK(P3))=CUMLD(P2)+LOAD(TRUCK(P3))-CUMLD(P3) 00054160
    NUMTRK=3                                              00054170
    TR(1)=P1                                             00054180
    TR(2)=P2                                             00054190
    TR(3)=P3                                             00054200
    GOTO 50                                              00054210
ENDIF
IF(LONER.EQ.1) THEN                               00054220
    IF(LOAD(TRUCK(P1))+CUMLD(P3)-CUMLD(P2).GT.TWTLIM) GOTO 13 00054230
    FEASLD(TRUCK(P1))=LOAD(TRUCK(P1))+CUMLD(P3)-CUMLD(P2) 00054240
    FEASLD(TRUCK(P2))=LOAD(TRUCK(P2))-CUMLD(P3)+CUMLD(P2) 00054250
    NUMTRK=2                                              00054260
    TR(1)=P1                                             00054270
    TR(2)=P2                                             00054280
    GOTO 50                                              00054290
ENDIF
IF(LONER.EQ.2) THEN                               00054300
    IF(LOAD(TRUCK(P3))+CUMLD(P2)-CUMLD(P1).GT.TWTLIM) GOTO 14 00054310
    FEASLD(TRUCK(P2))=LOAD(TRUCK(P2))-CUMLD(P2)+CUMLD(P1) 00054320
    FEASLD(TRUCK(P3))=LOAD(TRUCK(P3))+CUMLD(P2)-CUMLD(P1) 00054330
    NUMTRK=2                                              00054340
    TR(1)=P2                                             00054350
    TR(2)=P3                                             00054360
    GOTO 50                                              00054370
ENDIF
IF(LONER.EQ.3) THEN                               00054380
    IF(LOAD(TRUCK(P3))+CUMLD(P2)-CUMLD(P1).GT.TWTLIM) GOTO 14 00054390
    FEASLD(TRUCK(P2))=LOAD(TRUCK(P2))-CUMLD(P2)+CUMLD(P1) 00054400
    FEASLD(TRUCK(P3))=LOAD(TRUCK(P3))+CUMLD(P2)-CUMLD(P1) 00054410
    NUMTRK=2                                              00054420
    TR(1)=P2                                             00054430
    TR(2)=P3                                             00054440
    GOTO 50                                              00054450
ENDIF
IF(LONER.EQ.4) THEN                               00054460
    IF(LOAD(TRUCK(P2))+CUMLD(P1)-CUMLD(P3).GT.TWTLIM) GOTO 13 00054470
    FEASLD(TRUCK(P2))=LOAD(TRUCK(P2))+CUMLD(P1)-CUMLD(P3) 00054480
    FEASLD(TRUCK(P1))=LOAD(TRUCK(P1))-CUMLD(P1)+CUMLD(P3) 00054490
    NUMTRK=2                                              00054500
    TR(1)=P2                                             00054510
    TR(2)=P1                                             00054520
    GOTO 50                                              00054530
ENDIF
C
C
C      TYPE IV EXCHANGE - LOAD FEASIBILITY TEST      00054540
C
C
12 NTYPE=4                                         00054550
IF(CUMLD(P1)+CUMLD(P3).GT.TWTLIM) GOTO 13        00054560
    IF(LOAD(TRUCK(P1))-CUMLD(P1)+LOAD(TRUCK(P2))-CUMLD(P2).GT.TWTLIM) GOTO 13 00054570
*      GOTO 13                                         00054580
    IF(CUMLD(P2)+LOAD(TRUCK(P3))-CUMLD(P3).GT.TWTLIM) GOTO 13 00054590
    FEASLD(TRUCK(P1))=CUMLD(P1)+CUMLD(P3)                00054600
*      FEASLD(TRUCK(P2))=LOAD(TRUCK(P1))-CUMLD(P1)+LOAD(TRUCK(P2)) 00054610
*          -CUMLD(P2)                                00054620

```

```

      FEASLD(TRUCK(P3))=CUMLD(P2)+LOAD(TRUCK(P3))-CUMLD(P3)          00054680
      NUMTRK=3               00054690
      TR(1)=P1               00054700
      TR(2)=P2               00054710
      TR(3)=P3               00054720
      GOTO 50               00054730
C
C
C
C      TYPE V EXCHANGE - LOAD FEASIBILITY TEST
C
C
C      13 NTYPE=5
      IF(TRY(P1,P2).EQ.1) GOTO 14
      TRY(P1,P2)=1
      IF(LONER.EQ.3) GOTO 14
      IF(CUMLD(P1)+CUMLD(P2).GT.TWTLIM) GOTO 14
      IF(LOAD(TRUCK(P1))-CUMLD(P1)+LOAD(TRUCK(P2))-CUMLD(P2).GT.TWTLIM)
      * GOTO 14
      FEASLD(TRUCK(P1))=CUMLD(P1)+CUMLD(P2)
      FEASLD(TRUCK(P2))=LOAD(TRUCK(P1))-CUMLD(P1)+LOAD(TRUCK(P2))
      *           -CUMLD(P2)
      NUMTRK=2
      TR(1)=P1
      TR(2)=P2
      GOTO 50               00054740
                           00054750
                           00054760
                           00054770
                           00054780
                           00054790
                           00054800
                           00054810
                           00054820
                           00054830
                           00054840
                           00054850
                           00054860
                           00054870
                           00054880
                           00054890
                           00054900
                           00054910
                           00054920
                           00054930
C
C
C      TYPE VI EXCHANGE - LOAD FEASIBILITY TEST
C
C
C      14 NTYPE=6
      IF(TRY(P2,P3).EQ.1) GOTO 15
      TRY(P2,P3)=1
      IF(LONER.EQ.1) GOTO 15
      IF(CUMLD(P2)+CUMLD(P3).GT.TWTLIM) GOTO 15
      IF(LOAD(TRUCK(P2))-CUMLD(P2)+LOAD(TRUCK(P3))-CUMLD(P3).GT.TWTLIM)
      * GOTO 15
      FEASLD(TRUCK(P2))=CUMLD(P2)+CUMLD(P3)
      FEASLD(TRUCK(P3))=LOAD(TRUCK(P2))-CUMLD(P2)+LOAD(TRUCK(P3))
      *           -CUMLD(P3)
      NUMTRK=2
      TR(1)=P2
      TR(2)=P3
      GOTO 50               00054940
                           00054950
                           00054960
                           00054970
                           00054980
                           00054990
                           00055000
                           00055010
                           00055020
                           00055030
                           00055040
                           00055050
                           00055060
                           00055070
                           00055080
                           00055090
                           00055100
                           00055110
                           00055120
                           00055130
C
C
C      TYPE VII EXCHANGE - LOAD FEASIBILITY TEST
C
C
C      15 NTYPE=7
      IF(TRY(P1,P3).EQ.1) GOTO 8
      TRY(P1,P3)=1
      IF(LONER.EQ.2) GOTO 8
      IF(CUMLD(P1)+CUMLD(P3).GT.TWTLIM) GOTO 8
      IF(LOAD(TRUCK(P1))-CUMLD(P1)+LOAD(TRUCK(P3))-CUMLD(P3).GT.TWTLIM)
      * GOTO 8
      FEASLD(TRUCK(P1))=CUMLD(P1)+CUMLD(P3)
      FEASLD(TRUCK(P3))=LOAD(TRUCK(P1))-CUMLD(P1)+LOAD(TRUCK(P3))
      *           -CUMLD(P3)
      NUMTRK=2
      TR(1)=P1
      TR(2)=P3
      GOTO 50               00055140
                           00055150
                           00055160
                           00055170
                           00055180
                           00055190
                           00055200
                           00055210
                           00055220
                           00055230
                           00055240
                           00055250
                           00055260
                           00055270
                           00055280
                           00055290
                           00055300
                           00055310
                           00055320
                           00055330
                           00055340
                           00055350
                           00055360
                           00055370
                           00055380
C
C
C      DETERMINE WHETHER MAXIMUM LOAD DEVIATION HAS BEEN REDUCED.
C

```

```

C
      50 FMAXLD=-99          00055390
      FMINLD=999999          00055400
      DO 51 I=1,IROUTE       00055410
        IF(LOKK(DEPOT(I)).NE.0) GOTO 51
        IF(FEASLD(I).GT.FMAXLD) FMAXLD=FEASLD(I)
        IF(FEASLD(I).LT.FMINLD.AND.FEASLD(I).NE.0) FMINLD=FEASLD(I)
      51 CONTINUE             00055420
        IF(FMAXLD-FMINLD.GE.LDDEV+LDRlx) THEN
          FEASLD(TRUCK(P1))=LOAD(TRUCK(P1))
          FEASLD(TRUCK(P2))=LOAD(TRUCK(P2))
          FEASLD(TRUCK(P3))=LOAD(TRUCK(P3))
          IF(LONER.EQ.0) GOTO (10,11,12,13,14,15,8), NTYPE 00055430
          IF(LONER.EQ.1) GOTO (11,11,13,13,15,15,8), NTYPE 00055440
          IF(LONER.EQ.2) GOTO (11,11,13,13,14,8,8), NTYPE 00055450
          IF(LONER.EQ.3) GOTO (11,11,14,14,14,15,8), NTYPE 00055460
        ENDIF
C
C
C
C     ESTABLISH POTENTIAL ROUTE STRUCTURE RESULTING FROM EXCHANGE 00055470
C
C
C
C     IF(NTYPE.LE.4) CALL FXCH3(P1,P2,P3,FPRED,FSUCC,NTYPE) 00055480
C     IF(NTYPE.EQ.5) CALL FXCH2(P1,P2,FPRED,FSUCC,NULL) 00055490
C     IF(NTYPE.EQ.6) CALL FXCH2(P2,P3,FPRED,FSUCC,NULL) 00055500
C     IF(NTYPE.EQ.7) CALL FXCH2(P1,P3,FPRED,FSUCC,NULL) 00055510
C     IF(NTYPE.GT.4.AND.NULL.EQ.1) THEN 00055520
C       DO 49 I=1,NCITY+IROUTE 00055530
C         FPRED(I)=PRED(I)
C         FSUCC(I)=SUCC(I)
C       FEASLD(TRUCK(P1))=LOAD(TRUCK(P1)) 00055540
C       FEASLD(TRUCK(P2))=LOAD(TRUCK(P2)) 00055550
C       FEASLD(TRUCK(P3))=LOAD(TRUCK(P3)) 00055560
C       IF (LONER.EQ.0) GOTO (10,11,12,13,14,15,8), NTYPE 00055570
C       IF(LONER.EQ.1) GOTO (11,11,13,13,15,15,8), NTYPE 00055580
C       IF(LONER.EQ.2) GOTO (11,11,13,13,14,8,8), NTYPE 00055590
C       IF(LONER.EQ.3) GOTO (11,11,14,14,14,15,8),NTYPE 00055600
C     ENDIF
C
C
C
C     SOLVE TSP FOR EACH ROUTE AFFECTED BY THE EXCHANGE 00055610
C
C
C
C     DO 55 I=1,NUMTRK 00055620
C       ISTRT=DEPOT(TRUCK(TR(I)))
C       NEXT=FSUCC(ISTRT)
C     52 NODE=NEXT 00055630
C       IF(NODE.GT.NCITY) THEN 00055640
C         IEND=FPRED(NODE)
C         CALL TSP(ISTRT,IEND,FPRED,FSUCC,LNGTH,DIST,ALLOW) 00055650
C         IF(LNGTH.GT.DISTLM) GOTO 63
C         FEASLN(TRUCK(TR(I)))=LNGTH 00055660
C         GOTO 55
C       ENDIF
C       NEXT=FSUCC(NODE)
C       GOTO 52
C     55 CONTINUE 00055670
C
C
C
C     DETERMINE TOTAL DISTANCE INCREASE. 00055680
C
C
C
C     LTDIST=0 00055690
C     DO 60 I=1,IROUTE 00055700
C       60 LTDIST=LTDIST+FEASLN(I) 00055710
C
C
C

```

```

C      DETERMINE ROUTE LENGTH DEVIATION.          00056100
C                                              00056110
C                                              00056120
C
C      FMAXLN=-99                                00056130
C      FMINLN=999999                             00056140
DO 61 I=1,IROUTE
    IF(LOKK(DEPOT(I)).NE.0) GOTO 61           00056150
    IF(FEASLN(I).GT.FMAXLN) FMAXLN=FEASLN(I)
    IF(FEASLN(I).LT.FMINLN.AND.FEASLN(I).NE.0) FMINLN=FEASLN(I)
61 CONTINUE
C                                              00056190
C                                              00056200
C                                              00056210
C                                              00056220
C      TRADEOFF ANALYSIS.                      00056230
C                                              00056240
C                                              00056250
C
C      IF(LDRXL.GT.0) GOTO 62                  00056260
C      ELSE
IF(LTDIST.GT.DLIMIT.OR.FMAXLN-FMINLN.GT.LNDVLM) THEN
    NTRADE=NTRADE+1                          00056270
    IF(NTRADE.GT.500) NTRADE=500
    TRADE(1,NTRADE)=LDDEV-(FMAXLD-FMINLD)
    TRADE(1,NTRADE)=-TRADE(1,NTRADE)
    TRADE(2,NTRADE)=LTDIST-TDIST
    TRADE(3,NTRADE)=FMAXLN-FMINLN-LNDEV
    TRADE(4,NTRADE)=P1
    TRADE(5,NTRADE)=P2
    TRADE(6,NTRADE)=P3
    TRADE(7,NTRADE)=NTYPE
ENDIF
C                                              00056390
C                                              00056400
C                                              00056410
C                                              00056420
C
C      62 IF(LTDIST.GT.DLIMIT.OR.FMAXLN-FMINLN.GT.LNDVLM) THEN 00056430
C
C      REBUILD THE ROUTE STRUCTURE.            00056440
C                                              00056450
C
C      DO 64 I=NCITY+1,NCITY+IROUTE          00056460
    FSUCC(FPRED(I))=SUCC(FPRED(I))
64    FPRED(I)=PRED(I)
DO 66 I=1,NUMTRK
    FEASLD(TRUCK(TR(I)))=LOAD(TRUCK(TR(I)))
    FEASLN(TRUCK(TR(I)))=LENGTH(TRUCK(TR(I)))
    NODE=DEPOT(TRUCK(TR(I)))
    FPRED(NODE)=PRED(NODE)
    FSUCC(NODE)=SUCC(NODE)
    NEXT=SUCC(NODE)
65    NODE=NEXT
    IF(NODE.GT.NCITY) GOTO 66
    FPRED(NODE)=PRED(NODE)
    FSUCC(NODE)=SUCC(NODE)
    NEXT=SUCC(NODE)
    GOTO 65
66 CONTINUE
    IF(LONER.EQ.0) GOTO (10,11,12,13,14,15,8), NTTYPE
    IF(LONER.EQ.1) GOTO (11,11,13,13,15,15,8), NTTYPE
    IF(LONER.EQ.2) GOTO (11,11,13,13,14,8,8), NTTYPE
    IF(LONER.EQ.3) GOTO (11,11,14,14,14,15,8), NTTYPE
    CONTINUE
ENDIF
C                                              00056640
C                                              00056650
C                                              00056660
C                                              00056670
C                                              00056680
C                                              00056690
C                                              00056700
C                                              00056710
C                                              00056720
C                                              00056730
C                                              00056740
C      CHANGE ROUTE STRUCTURE                 00056750
C                                              00056760
C                                              00056770
C
MAXLN=FMAXLN                                     00056780
MINLN=FMINLN                                     00056790
MAXLD=FMAXLD                                     00056800

```

```

MINLD=FMINLD          00056810
LNDEV=MAXLN-MINLN    00056820
LDDEV=MAXLD-MINLD    00056830
DO 67 I=1,NCITY+IROUTE 00056840
  PRED(I)=FPRED(I)    00056850
67   SUCC(I)=FSUCC(I) 00056860
  NEXT=NCITY+1         00056870
  INSIDE=0             00056880
68 NODE=NEXT          00056890
  IF(NODE.EQ.NCITY+1.AND.INSIDE.EQ.1) GOTO 69 00056900
  INSIDE=1             00056910
  IF(NODE.GT.NCITY) THEN 00056920
    ITRK=TRUCK(NODE)    00056930
    ILD=0                00056940
    ILN=0                00056950
  ENDIF
  ILD=ILD+DEMAND(NODE) 00056960
  IF(NODE.LE.NCITY) ILN=ILN+DIST(NODE,PRED(NODE))+ALLOW 00056970
  CUMLD(NODE)=ILD      00056980
  CUMLN(NODE)=ILN      00056990
  TRUCK(NODE)=ITRK     00057000
  FEASLD(ITRK)=ILD     00057010
  NEXT=SUCC(NODE)       00057020
  IF(NEXT.GT.NCITY) FEASLN(ITRK)=ILN+DIST(NODE,NEXT) 00057030
  GOTO 68               00057040
69 TDIST=0              00057050
  TLOAD=0               00057060
  DO 70 I=1,IROUTE      00057070
    LOAD(I)=FEASLD(I)    00057080
    LENGTH(I)=FEASLN(I)  00057090
    TLOAD=TLOAD+LOAD(I)  00057100
70   TDIST=TDIST+LENGTH(I) 00057110
00057120
C
C
C
C   ROTATE              00057130
C
C
C
TWLIM=MINO(WTLIM,MAXLD+LDDEV) 00057140
IRLX=LDRlx                00057150
LDRlx=0                     00057160
NTRADE=0                    00057170
ISTART=END                  00057180
IF(IRLX.GT.0) ISTART=DEPOT(TRUCK(PRED(ISTART))) 00057190
END=PRED(ISTART)            00057200
START=ISTART                00057210
P1=END                      00057220
DO 71 I=1,NCITY+IROUTE      00057230
  DO 71 J=1,NCITY+IROUTE      00057240
71   TRY(I,J)=0              00057250
  GOTO 6                      00057260
END
C
C
C*****
C
C   SUBROUTINE NONDOM(ITRADE,NT) 00057330
C
C
C   THIS SUBROUTINE ELIMINATES ALL DOMINATED TRADEOFFS FROM THE SET 00057340
C   OF TRADEOFFS.           00057350
C
C*****
C
C   DIMENSION ITRADE(7,500)    00057360
C
C
C   IF(NT.LE.1) RETURN          00057370
N=1                         00057380
DO 10 I=2,NT                  00057390
  N1=N                         00057400

```

```

      DO 9 K=1,N1          00057520
      *      IF(ITRADE(2,I).GE.ITRADE(2,K).AND.ITRADE(3,I).GE.ITRADE(3,K)) 00057530
      *      GOTO 10          00057540
9   CONTINUE          00057550
N=N+1          00057560
ITRADE(1,N)=ITRADE(1,I)          00057570
ITRADE(2,N)=ITRADE(2,I)          00057580
ITRADE(3,N)=ITRADE(3,I)          00057590
ITRADE(4,N)=ITRADE(4,I)          00057600
ITRADE(5,N)=ITRADE(5,I)          00057610
ITRADE(6,N)=ITRADE(6,I)          00057620
ITRADE(7,N)=ITRADE(7,I)          00057630
10 CONTINUE          00057640
NT=N          00057650
RETURN          00057660
END          00057670
C          00057680
C          00057690
C*****          00057700
C          00057710
C      SUBROUTINE DISPLA(NUMBER,IROUTE,NCITY,PNAME,XCOORD,YCOORD,TIME) 00057720
C          00057730
C          00057740
C      THIS SUBROUTINE IS USED TO DISPLAY A PRIOR SOLUTION TO 00057750
C      THE WORKLOAD-BALANCED VEHICLE ROUTING PROBLEM.          00057760
C          00057770
C*****          00057780
CHARACTER*16 OBJ(12)          00057790
CHARACTER*44 PNAME          00057800
INTEGER PRED(12,120),SUCC(12,120),TRUCK(12,120),CUMLN(12,120), 00057810
*CUMLD(12,120),DEPOT(12,20),LOAD(12,20),LENGTH(12,20),MAXLN(12), 00057820
*MINLN(12),MAXLD(12),MINLD(12),ITRADE(12,7,500),LDDVLM(12), 00057830
*LNDVLM(12),DLIMIT(12),NT(12),RTSIZE(20),XCOORD(0:120), 00057840
*YCOORD(0:120),LOKK(12,120),TDIST          00057850
DIMENSION XCENTR(20),YCENTR(20)          00057860
C          00057870
C      COMMON /STORE/ PRED,SUCC,TRUCK,CUMLN,CUMLD,DEPOT,LOAD,LENGTH, 00057880
*MAXLN,MINLN,MAXLD,MINLD,ITRADE,NT,LDDVLM,LNDVLM,DLIMIT,LOKK,OBJ 00057890
C          00057900
C          00057910
C          00057920
C      CALCULATE TOTAL DISTANCE, LOAD DEVIATION, AND LENGTH DEVIATION. 00057930
C          00057940
C          00057950
TDIST=0          00057960
DO 1 I=1,IROUTE          00057970
1   TDIST=TDIST+LENGTH(NUMBER,I)          00057980
LDDEV=MAXLD(NUMBER)-MINLD(NUMBER)          00057990
LNDEV=MAXLN(NUMBER)-MINLN(NUMBER)          00058000
C          00058010
C          00058020
C          00058030
C      DISPLAY ROUTES ON GRAPHICS SCREEN          00058040
C          00058050
C          00058060
CALL NEWPAG          00058070
CALL TWINDO(0,4095,0,3120)          00058080
CALL MOVABS(0,0)          00058090
CALL DRWABS(4095,0)          00058100
CALL DRWABS(4095,3120)          00058110
CALL DRWABS(0,3120)          00058120
CALL DRWABS(0,0)          00058130
CALL MOVABS(1030,0)          00058140
CALL DRWABS(1030,3120)          00058150
CALL MOVABS(1030,2800)          00058160
CALL DRWABS(4095,2800)          00058170
CALL MOVABS(1130,2950)          00058180
CALL AOUTST(44,PNAME)          00058190
CALL TWINDO(1030,4095,0,2800)          00058200
DO 2 J=1,IROUTE          00058210
      RTSIZE(J)=0          00058220

```

```

      XCENTR(J)=0.0          00058230
2     YCENTR(J)=0.0          00058240
NEXT=NCITY+1          00058250
X=XCOORD(NEXT)          00058260
Y=YCOORD(NEXT)          00058270
XCENTR(TRUCK(NUMBER,NEXT))=XCENTR(TRUCK(NUMBER,NEXT))+X 00058280
YCENTR(TRUCK(NUMBER,NEXT))=YCENTR(TRUCK(NUMBER,NEXT))+Y 00058290
RTSIZE(TRUCK(NUMBER,NEXT))=RTSIZE(TRUCK(NUMBER,NEXT))+1 00058300
CALL MOVEA(X,Y)          00058310
CALL MOVREL(40,0)          00058320
CALL DRWREL(0,40)          00058330
CALL DRWREL(-80,0)          00058340
CALL DRWREL(0,-80)          00058350
CALL DRWREL(80,0)          00058360
CALL DRWREL(0,40)          00058370
CALL MOVEA(X,Y)          00058380
INSIDE=0          00058390
3     NODE=NEXT          00058400
IF(NODE.EQ.NCITY+1.AND.INSIDE.EQ.1) GOTO 4          00058410
INSIDE=1          00058420
X=XCOORD(NODE)          00058430
Y=YCOORD(NODE)          00058440
XCENTR(TRUCK(NUMBER,NODE))=XCENTR(TRUCK(NUMBER,NODE))+X 00058450
YCENTR(TRUCK(NUMBER,NODE))=YCENTR(TRUCK(NUMBER,NODE))+Y 00058460
RTSIZE(TRUCK(NUMBER,NODE))=RTSIZE(TRUCK(NUMBER,NODE))+1 00058470
CALL DRAWA(X,Y)          00058480
ICHR1=NODE/100          00058490
ICHR2=NODE/10 - ICHR1*10          00058500
ICHR3=NODE - (ICHR1*100+ICHR2*10)          00058510
IF(NODE.LE.NCITY) THEN          00058520
  CALL MOVREL(20,0)          00058530
  CALL DRWREL(0,20)          00058540
  CALL DRWREL(-40,0)          00058550
  CALL DRWREL(0,-40)          00058560
  CALL DRWREL(40,0)          00058570
  CALL DRWREL(0,20)          00058580
ENDIF          00058590
CALL MOVEA(X,Y)          00058600
NEXT=SUCC(NUMBER,NODE)          00058610
GOTO 3          00058620
4     CONTINUE          00058630
X=XCOORD(NODE)          00058640
Y=YCOORD(NODE)          00058650
CALL DRAWA(X,Y)          00058660
DO 5 J=1,IROUTE          00058670
  IF(LOAD(NUMBER,J).GT.0) THEN          00058680
    XCENTR(J)=XCENTR(J)/FLOAT(RTSIZE(J))          00058690
    YCENTR(J)=YCENTR(J)/FLOAT(RTSIZE(J))          00058700
    CALL MOVEA(XCENTR(J),YCENTR(J))          00058710
    ICHR1=J/10          00058720
    ICHR2=J-ICHR1*10          00058730
    IF(ICHR1.NE.0) CALL ANCHO(ICHR1+48)          00058740
    CALL ANCHO(ICHR2+48)          00058750
  ENDIF          00058760
5     CONTINUE          00058770
C          00058780
C          00058790
C          00058800
C     DISPLAY SOLUTION STATISTICS          00058810
C          00058820
C          00058830
CALL HOME          00058840
CALL ANMODE          00058850
WRITE(6,100) NUMBER,OBJ(NUMBER)          00058860
100 FORMAT(1H //1X,'SOLUTION NUMBER',I2//1X,A/' MINIMIZATION PROBLEM' 00058870
*// ' ROUTE LOAD LENGTH')
DO 6 II=1,IROUTE          00058880
  IF(LOAD(NUMBER,IROUTE).LT.0) GOTO 5          00058890
  WRITE(6,101) II,LOAD(NUMBER,II),LENGTH(NUMBER,II)          00058900
6     CONTINUE          00058910
101 FORMAT(I4,I8,1X,I5)          00058920
                                00058930

```

```

      WRITE(6,102) TDIST,LDDEV,LNDEV          00058940
102 FORMAT(' TOT. DIST =',I6,' LOAD DEV. = ',I5,' LENGTH DEV. =',I4) 00058950
      CALL ELAPSE(ITIME)
      TIME=TIME+FLOAT(ITIME)/1000.
      WRITE(6,103) TIME                      00058960
103 FORMAT(1H , 'CPU SECONDS:',F6.2)       00058970
      WRITE(6,104)
104 FORMAT(1H ,/////////, ' HIT <RTN> TO CONTINUE')
      CALL TINPUT(MMM)
      RETURN
      END

C
C
C
C*****SUBROUTINE BKTRAK(NUMBER,ITRADE,NT,OBJ,OLDOBJ,LIMIT1,LIMIT2, 00059080
*           CNSTR1,CNSTR2)                                     00059090
C
C
C   THIS SUBROUTINE IS USED TO BACKTRACK TO A PRIOR SOLUTION OF THE 00059100
C   WORKLOAD-BALANCED VEHICLE ROUTING PROBLEM.                00059110
C
C*****CHARACTER*44 PNAME,IPLACE                         00059120
CHARACTER*16 OBJ,STOBJ(12),CNSTR1,CNSTR2             00059130
CHARACTER*1 ANSWER                                 00059140
INTEGER EUCLID,CITY,XCOORD(O:120),YCOORD(O:120),DEMAND(O:120) 00059150
INTEGER HEAD(120),TAIL(120),PRED(120),SUCC(120),ROUTES,TWGT,WTLIM 00059160
INTEGER DIST,ALLOW,TDIST,DISTLM,TRUCK,IR(100),IFLAG(40),        00059170
* PERMI(40),PERMJ(40)                                00059180
DOUBLE PRECISION DSEED                           00059190
INTEGER START,END,POINT1,POINT2,D,FEASLD(20),FEASLN(20) 00059200
INTEGER FTRUCK,FWD,BACK,TEMTRK(120),CUMLD(120),CUMLN(120) 00059210
INTEGER FOUND,TRCNT,TRK,TAG,D1,D2,D3,D4,D5,D6,D7,D8 00059220
INTEGER FSTART,FEND,FPRED(120),FSUCC(120),BACTRT,DSTRXLX 00059230
INTEGER PERMPR(120),PERMSU(120),PERMTR(120),RTSIZE(20) 00059240
INTEGER BESTRT,BESTDS,BCUMLN(120),BCUMLD(120),BESTP(120) 00059250
INTEGER BESTS(120),BESTLN(20),BESTLD(20),BDEPOT(20),BESTTR(120) 00059260
INTEGER DLIMIT,TARRAY(40),DEPOT(20),CLRNDX(12),BLDDEV,BLNDEV 00059270
INTEGER BMAXLD,BMINLD,BMAXLN,BMINLN,LOKK(120),STLOKK(12,120) 00059280
INTEGER BNTRAD,ITRADE(7,500),OLDOBJ                 00059290
INTEGER SOLNO,STPRED(12,120),STSUCC(12:120),STTRK(12,120), 00059300
*STCULN(12,120),STCULD(12,120),STDEP(12,20),STLOAD(12,20), 00059310
*STLNTH(12,20),STMXLN(12),STMNLN(12),STMXLD(12),STMNLD(12), 00059320
*STITRD(12,7,500),STLDVL(12),STLNVL(12),STDLIM(12),STNT(12) 00059330
DIMENSION BTRADE(7,500),TRADE(7,500),WK(14)          00059340
DIMENSION DIST(O:120,O:120),SAVING(3,6000),SORT(6000),PERMSV(40) 00059350
DIMENSION ISORT(6000),JSORT(6000),LOAD(120),TRUCK(120) 00059360
DIMENSION LENGTH(120),WORK(6),XCENTR(20),YCENTR(20) 00059370
COMMON IROUTE,NCITY,IRUN,START,END,WTLIM,DISTLM,ALLOW,NPERM, 00059380
*ICOUNT,LDDVLM,LNDVLM,DLIMIT,MAXLD,MINLD,MAXLN,MINLN,D1,D2,D3,D4, 00059390
*D5,D6,DEPOT,PRED,SUCC,TRUCK,DEMAND,LENGTH,LOAD,CUMLD,CUMLN,TEMTRK, 00059400
*FEASLD,FEASLN,PERMPR,PERMSU,PERMTR,ISORT,JSORT,SORT,DIST,XCOORD, 00059410
*YCOORD,LOKK,TRADE,NTRADE,DSEED                   00059420
COMMON /STORE/STPRED,STSUCC,STTRK,STCULN,STCULD,STDEP,STLOAD, 00059430
*STLNTH,STMXLN,STMNLN,STMXLD,STMNLD,STITRD,STNT,STLDVL,STLNVL, 00059440
*STDLIM,STLOKK,STOBJ                            00059450
C
C
C
C   RESET ALL PROBLEM PARAMETERS TO PRIOR VALUES. 00059460
C
C
C
DO 1 I=1,NCITY+IROUTE
      PRED(I)=STPRED(NUMBER,I)                  00059470
      SUCC(I)=STSUCC(NUMBER,I)                  00059480
      LOKK(I)=STLOKK(NUMBER,I)                  00059490
      TRUCK(I)=STTRK(NUMBER,I)                  00059500
      CUMLN(I)=STCULN(NUMBER,I)                  00059510
1     CUMLD(I)=STCULD(NUMBER,I)                  00059520
                                         00059530
                                         00059540
                                         00059550
                                         00059560
                                         00059570
                                         00059580
                                         00059590
                                         00059600
                                         00059610
                                         00059620
                                         00059630
                                         00059640

```

```

DO 2 I=1,IROUTE                               00059650
  DEPOT(I)=STDEP(NUMBER,I)                   00059660
  LOAD(I)=STLOAD(NUMBER,I)                  00059670
2   LENGTH(I)=STLNTH(NUMBER,I)               00059680
  MAXLN=STMXLN(NUMBER)                     00059690
  MINLN=STMNLN(NUMBER)                     00059700
  MAXLD=STMXLD(NUMBER)                     00059710
  MINLD=STMNLD(NUMBER)                     00059720
  NT=STNT(NUMBER)                         00059730
  DO 3 I=1,NT                                00059740
    DO 3 J=1,7                                00059750
3   ITRADE(J,I)=STITRD(NUMBER,J,I)          00059760
    LDDVLM=STLDVL(NUMBER)                   00059770
    LNDVLM=STLNVL(NUMBER)                   00059780
    DLIMIT=STDLIM(NUMBER)                   00059790
    OBJ=STOBJ(NUMBER)                      00059800
    IF(OBJ.EQ.'TOTAL DISTANCE') THEN       00059810
      LIMIT1=LDDVLM                         00059820
      LIMIT2=LNDVLM                         00059830
      CNSTR1='LOAD DEVIATION'              00059840
      CNSTR2='LENGTH DEVIATION'            00059850
      OLDOBJ=1                             00059860
      RETURN                                00059870
    ENDIF
    IF(OBJ.EQ.'LOAD DEVIATION') THEN       00059890
      LIMIT1=DLIMIT                         00059900
      LIMIT2=LNDVLM                         00059910
      CNSTR1='TOTAL DISTANCE'              00059920
      CNSTR2='LENGTH DEVIATION'            00059930
      OLDOBJ=2                             00059940
      RETURN                                00059950
    ENDIF
    IF(OBJ.EQ.'LENGTH DEVIATION') THEN     00059960
      LIMIT1=DLIMIT                         00059970
      LIMIT2=LDDVLM                         00059980
      CNSTR1='TOTAL DISTANCE'              00060000
      CNSTR2='LOAD DEVIATION'              00060010
      OLDOBJ=3                             00060020
      RETURN                                00060030
    ENDIF
    END
C
C
C
C*****SUBROUTINE ADJUST(NUMBER)           00060080
C
C
C   THIS SUBROUTINE ACCEPTS MANUAL ADJUSTMENTS TO A VEHICLE ROUTE AND 00060140
C   EVALUATES THE IMPACT OF THOSE CHANGES ON THE LENGTH OF THE ALTERED 00060150
C   ROUTE.                                     00060160
C
C
C*****DOUBLE PRECISION DSEED             00060170
C   INTEGER START,END,WTLIM,DISTLM,ALLOW,DLIMIT,D1,D2,D3,D4,D5,D6, 00060220
C   *DEPOT(20),PRED(120),SUCC(120),TRUCK(120),DEMAND(0:120),LENGTH(120) 00060230
C   *,LOAD(120),CUMLD(120),CUMLN(120),TEMTRK(120),FEASLD(20),FEASLN(20) 00060240
C   *,PERMPR(120),PERMSU(120),PERMTR(120),ISORT(6000),JSORT(6000),    00060250
C   *DIST(0:120,0:120),XCOORD(0:120),YCOORD(0:120),LOKK(120)        00060260
C   INTEGER XSTART,XEND,XPRED(120),XSUCC(120),XCUMLD(120),XCUMLN(120),00060270
C   *RTSIZE(20)                           00060280
C   DIMENSION TRADE(7,500),XCENTR(20),YCENTR(20),SORT(6000)          00060290
C   COMMON IROUTE,NCITY,IRUN,START,END,WTLIM,DISTLM,ALLOW,NPERM,        00060300
C   *ICOUNT,LDDVLM,LNDVLM,DLIMIT,MAXLD,MINLD,MAXLN,MINLN,D1,D2,D3,D4, 00060310
C   *D5,D6,DEPOT,PRED,SUCC,TRUCK,DEMAND,LENGTH,LOAD,CUMLD,CUMLN,TEMTRK,00060320
C   *FEASLD,FEASLN,PERMPR,PERMSU,PERMTR,ISORT,JSORT,SORT,DIST,XCOORD, 00060330
C   *YCOORD,LOKK,TRADE,NTRADE,DSEED          00060340
C
C                                         00060350

```

```

C          00060360
C      SET UP TEMPORARY VARIABLES AND ARRAYS. 00060370
C          00060380
C          00060390
C          00060400
C          00060410
C          00060420
10     CONTINUE 00060430
DO 20 I=1,IRROUTE 00060440
    IF(TRUCK(PRED(DEPOT(I))).EQ.NUMBER) XEND=DEPOT(I) 00060450
    XPRED(I)=PRED(I)
    XSUCC(I)=SUCC(I)
    XCUMLD(I)=CUMLD(I)
20     XCUMLN(I)=CUMLN(I) 00060460
C          00060470
C          00060480
C          00060490
C          00060500
C          00060510
C      DEFINE VIRTUAL WINDOW AND SCREEN WINDOW FOR ROUTE TO BE ADJUSTED. 00060520
C          00060530
C          00060540
C          00060550
C          00060560
C          00060570
C          00060580
C          00060590
MAXX=-99999 00060600
MINX=999999 00060610
MAXY=-99999 00060620
MINY=999999 00060630
DO 30 I=1,NCITY+IRROUTE 00060640
    IF(TRUCK(I).NE.NUMBER) GOTO 30
    IF(XCOORD(I).GT.MAXX) MAXX=XCOORD(I)
    IF(XCOORD(I).LT.MINX) MINX=XCOORD(I)
    IF(YCOORD(I).GT.MAXY) MAXY=YCOORD(I)
    IF(YCOORD(I).LT.MINY) MINY=YCOORD(I)
30     CONTINUE 00060650
LIM=MAXO(MAXX-MINX,MAXY-MINY) 00060660
X1=MINX-10 00060670
X2=X1+FLOAT(LIM)+20. 00060680
Y1=MINY-10 00060690
Y2=Y1+FLOAT(LIM)+20. 00060700
CALL DWINDO(X1,X2,Y1,Y2) 00060710
CALL TWINDO(1030,4095,0,2800) 00060720
C          00060730
C          00060740
C          00060750
C          00060760
C          00060770
C          00060780
C      DISPLAY ROUTE TO BE ADJUSTED.
C      ;
C          00060790
CALL NEWPAG 00060800
NEXT=DEPOT(NUMBER) 00060810
DO 40 J=1,IRROUTE 00060820
    RTSIZE(J)=0 00060830
    XCENTR(J)=0.0 00060840
40     YCENTR(J)=0.0 00060850
X=XCOORD(NEXT) 00060860
Y=YCOORD(NEXT) 00060870
XCENTR(TRUCK(NEXT))=XCENTR(TRUCK(NEXT)) + X 00060880
YCENTR(TRUCK(NEXT))=YCENTR(TRUCK(NEXT)) + Y 00060890
RTSIZE(TRUCK(NEXT))=RTSIZE(TRUCK(NEXT)) + 1 00060900
CALL MOVEA(X,Y) 00060910
CALL MOVREL(40,0) 00060920
CALL DRWREL(0,40) 00060930
CALL DRWREL(-80,0) 00060940
CALL DRWREL(0,-80) 00060950
CALL DRWREL(80,0) 00060960
CALL DRWREL(0,40) 00060970
CALL MOVEA(X,Y) 00060980
INSIDE=0 00060990
41     NODE=NEXT 00061000
    IF(NODE.GE.NCITY+1.AND.INSIDE.EQ.1) GOTO 42
    INSIDE=1 00061010
    X=XCOORD(NODE) 00061020
    Y=YCOORD(NODE) 00061030
    XCENTR(TRUCK(NODE))=XCENTR(TRUCK(NODE)) + X 00061040
    YCENTR(TRUCK(NODE))=YCENTR(TRUCK(NODE)) + Y 00061050
    RTSIZE(TRUCK(NODE))=RTSIZE(TRUCK(NODE)) + 1 00061060

```

```

CALL DRAWA(X,Y)          00061070
ICHR1=NODE/100           00061080
ICHR2=NODE/10 - ICHR1*10 00061090
ICHR3=NODE - (ICHR1*100 + ICHR2*10) 00061100
IF(NODE.LE.NCITY) THEN   00061110
  CALL MOVREL(20,0)      00061120
  CALL DRWREL(0,20)      00061130
  CALL DRWREL(-40,0)     00061140
  CALL DRWREL(0,-40)     00061150
  CALL DRWREL(40,0)      00061160
  CALL DRWREL(0,20)      00061170
  CALL ANCHO(ICHR1+48)   00061180
  CALL ANCHO(ICHR2+48)   00061190
  CALL ANCHO(ICHR3+48)   00061200
ENDIF                   00061210
CALL MOVEA(X,Y)          00061220
NEXT=SUCC(NODE)          00061230
GOTO 41                  00061240
42 CONTINUE               00061250
X=XCOORD(NODE)           00061260
Y=YCOORD(NODE)           00061270
CALL DRAWA(X,Y)           00061280
DO 43 J=NUMBER,NUMBER    00061290
  IF(LOAD(J).GT.0) THEN   00061300
    XCENTR(J)=XCENTR(J)/FLOAT(RTSIZE(J)) 00061310
    YCENTR(J)=YCENTR(J)/FLOAT(RTSIZE(J)) 00061320
    CALL MOVEA(XCENTR(J),YCENTR(J))        00061330
    ICHR1=J/10                  00061340
    ICHR2=J-ICHR1*10            00061350
    IF(ICHR1.NE.0) CALL ANCHO(ICHR1+48) 00061360
    CALL ANCHO(ICHR2+48)       00061370
  ENDIF                   00061380
43 CONTINUE               00061390
C                         00061400
C                         00061410
C                         00061420
C                         READ IN CHANGES TO THE ROUTE. 00061430
C                         00061440
C                         00061450
C                         CALL HOME          00061460
C                         CALL ANMODE         00061470
C                         NEXT=XSTART        00061480
C                         WRITE(6,100)       00061490
44 NODE=NEXT              00061500
  READ(5,*) I              00061510
  IF(I.EQ.999) GOTO 50      00061520
C                         ELSE             00061530
  XSUCC(NODE)=I            00061540
  XPRED(I)=NODE            00061550
  NEXT=I                   00061560
  GOTO 44                  00061570
50 XSUCC(NODE)=XEND        00061580
  XPRED(XEND)=NODE          00061590
C                         00061600
C                         CALCULATE EFFECT OF CHANGES. 00061610
C                         00061620
C                         00061630
C                         00061640
C                         ILD=0             00061650
C                         ILN=0             00061660
C                         NEXT=XSTART        00061670
60 NODE=NEXT              00061680
  IF(NODE.EQ.XEND) GOTO 70 00061690
  ILD=ILD+DEMAND(NODE)    00061700
  IF(NODE.LE.NCITY) ILN=ILN+DIST(NODE,XPRED(NODE))+ALLOW 00061710
  XCUMLD(NODE)=ILD          00061720
  XCUMLN(NODE)=ILN          00061730
  NEXT=XSUCC(NODE)          00061740
  GOTO 60                  00061750
70 ILN=ILN+DIST(NODE,XPRED(NODE)) 00061760
  IF(ILN.GE.LENGTH(NUMBER)) THEN 00061770

```

```

CALL ANMODE          00061780
WRITE(6,102) ILN,LENGTH(NUMBER) 00061790
CALL TINPUT(MMMM) 00061800
RETURN             00061810
ENDIF              00061820
IF(ILN.LT.LENGTH(NUMBER)) THEN 00061830
  CALL ANMODE          00061840
  WRITE(6,103) ILN,LENGTH(NUMBER) 00061850
  LENGTH(NUMBER)=ILN           00061860
  DO 71 I=1,NCITY+IROUTE      00061870
    PRED(I)=XPRED(I)          00061880
    SUCC(I)=XSUCC(I)          00061890
    CUMLD(I)=XCUMLD(I)        00061900
  71   CUMLN(I)=XCUMLN(I)      00061910
    MAXLN=-999999               00061920
    MINLN=999999                00061930
    DO 72 I=1,IROUTE          00061940
      IF(LOKK(DEPOT(I)).EQ.1) GOTO 72 00061950
      IF(LENGTH(I).LT.MINLN.AND.LENGTH(I).NE.0) MINLN=LENGTH(I) 00061960
      IF(LENGTH(I).GT.MAXLN) MAXLN=LENGTH(I) 00061970
  72   CONTINUE             00061980
    CALL TINPUT(MMMM)          00061990
    RETURN                      00062000
  ENDIF
100 FORMAT(1H ,'ENTER NODES, IN ORDER,'// END INPUT WITH 999// ?') 00062020
102 FORMAT(1H ,'NEW ROUTE LENGTH =',I4// 'OLD ROUTE LENGTH =',I4// NO I00062030
  *IMPROVEMENT -- CHANGES NOT IMPLEMENTED// HIT <RTN> TO CONTINUE') 00062040
103 FORMAT(1H ,'NEW ROUTE LENGTH =',I4// 'OLD ROUTE LENGTH =',I4// IMPRO0062050
  *OVEMENT IN ROUTE LENGTH -- CHANGES IMPLEMENTED// HIT <RTN> TO CON00062060
  *TINUE')
  END                         00062070
C                               00062080
C                               00062090
C                               00062100
C*****SUBROUTINE LOKK (NUMBER,LOKK,NCITY,TRUCK,IROUTE) 00062110
C                               00062120
C                               00062130
C                               00062140
C                               00062150
C THIS SUBROUTINE 'LOCKS OUT' (I.E., MAKES UNAVAILABLE FOR 00062160
C CALCULATIONS OF ROUTE-LOAD AND ROUTE-LENGTH DEVIATION) A VEHICLE 00062170
C ROUTE.                  00062180
C                               00062190
C*****INTEGER TRUCK(120),LOKK(120) 00062200
C                               00062210
C                               00062220
C                               00062230
C                               00062240
C DO 10 I=1,NCITY+IROUTE      00062250
C   IF(TRUCK(I).EQ.NUMBER) LOKK(I)=1 00062260
  10 CONTINUE                 00062270
  RETURN                      00062280
  END                         00062290

```

2

VITA

Jerry Denver Allison

Candidate for the Degree of

Doctor of Philosophy

Thesis: WORKLOAD BALANCING IN VEHICLE ROUTING PROBLEMS

Major Field: Industrial Engineering and Management

Biographical:

Personal Data: Born in Mineral Wells, Texas, February 12, 1944, the son of Joseph Earl and Desla Frances Allison. Married to Susan Carol Fisk on August 19, 1978.

Education: Graduated from Weatherford High School, Weatherford, Texas, in May, 1962; received Bachelor of Science Degree in Industrial Engineering from the University of Texas at Arlington in January, 1968; received Master of Engineering Degree in Industrial Engineering from Texas A and M University in May, 1970; completed requirements for the Doctor of Philosophy degree at Oklahoma State University in December, 1986.

Professional Experience: General Engineer, U.S. Army Materiel Command, Rock Island, Illinois, February, 1968 to September, 1970; Senior Project Engineer, Mason and Hanger, Amarillo, Texas, October, 1971 to August, 1981; Teaching and Research Associate, School of Industrial Engineering and Management, Oklahoma State University, September, 1981 to June, 1986.

Professional Organizations: Institute of Industrial Engineers, Alpha Pi Mu, The Institute of Management Sciences.