A VEHICLE ASSIGNMENT PROBLEM

ALGORITHM

A THESIS

Presented to

The Faculty of the Graduate Division

by

John Edgar Felch, Jr.

In Partial Fulfillment

of the Requirements for the Degree

Master of Science in Operations Research

Georgia Institute of Technology

June, 1974
A VEHICLE ASSIGNMENT PROBLEM

ALGORITHM

Approved:

\underline{C. M. Shetty, Chairman}

\underline{Richard H. Deane}

\underline{Donovan B. Young}

Date approved by Chairman: \text{May 16, 1974}
ACKNOWLEDGMENTS

The contribution of Professor C. M. Shetty to the completion of this paper cannot be easily expressed. His keen insight and patience during long discussions is greatly appreciated. His guidance, instruction, and inspiration will remain a most significant part of my academic experience.

I would also like to thank the members of the reading committee, Dr. D. B. Young and Dr. R. H. Deane, for their willingness to give of their valuable time in reviewing and evaluating this work. The programming assistance of Dr. R. L. Rardin is also acknowledged with grateful appreciation.

Finally, I would like to formally thank my wife, Judy, for her encouragement, understanding, and faith. Her selfless typing of the drafts of this thesis made the process much smoother than would have otherwise been possible. And most important, her management of everything else allowed me to devote my time to this research.
# TABLE OF CONTENTS

Page

ACKNOWLEDGMENTS ........................................... ii

LIST OF TABLES ........................................... iv

LIST OF ILLUSTRATIONS ..................................... v

SUMMARY .................................................. vi

Chapter

I. INTRODUCTION ........................................... 1

   Literature Review
   Problem Statement and Mathematical Model

II. SOLUTION PROCEDURE .................................... 19

   The Algorithm
   Integer Solution
   Further Analysis of Solution
   An Example

III. COMPUTATIONAL RESULTS AND CONCLUSIONS .............. 38

   Test Problems
   Conclusions
   Recommendations

APPENDIX

A .......................................................... 46

B .......................................................... 65

BIBLIOGRAPHY ............................................ 67
# LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Literature Synopsis</td>
<td>8</td>
</tr>
<tr>
<td>2. Comparison of Multiple-Depot Formulations</td>
<td>9</td>
</tr>
<tr>
<td>3. Notation</td>
<td>12</td>
</tr>
<tr>
<td>4. Test Problem Solutions</td>
<td>43</td>
</tr>
</tbody>
</table>
# LIST OF ILLUSTRATIONS

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Illustration of case (i) of Theorem 1</td>
<td>23</td>
</tr>
<tr>
<td>2.</td>
<td>Illustration of case (iii) of Theorem 1</td>
<td>24</td>
</tr>
<tr>
<td>3.</td>
<td>Illustration of case (ii) of Theorem 1</td>
<td>24</td>
</tr>
<tr>
<td>4.</td>
<td>Flow Diagram of the Algorithm (for continuous solutions)</td>
<td>39</td>
</tr>
</tbody>
</table>
SUMMARY

This study deals with the problem of assigning vehicles, available at several depots, to meet the requirement for vehicles at several job sites. A vehicle may perform more than one job after it leaves the depot, but must return to the originating depot. Each job requirement essentially states the number and type of vehicle required, the time required, and the release time.

The model for the above problem involves a large number of variables and equations and would be beyond the capacity of present computers. The strategy used is first to identify null variables based on the dynamic constraints of the problem. Then the resulting static problem is solved by Dantzig-Wolfe decomposition where each subproblem turns out to be a shortest path problem for which efficient network codes are available. Finally, the resulting solution is integerized if not already so. This last phase is heuristic but easily implemented.

The algorithm is applied to some test problems and the results are encouraging. Larger problems are not solved because of memory storage restrictions. No attempt has been made to optimize the program. However, the study does discuss the size of problems that one can expect to handle if a more efficient means of storing information is adopted.
CHAPTER I

INTRODUCTION

Although operations research has made significant contributions to all phases of the transportation industry over the last twenty years, we shall deal in this paper only with the particular aspects of the transportation field associated with vehicle fleet management. The problems considered in this study are of interest both academically and commercially. Academically, the problem is related to other problems in scheduling, network theory, and assembly-line balancing. Obvious commercial value is achieved by any successful effort which results in cost reduction or improved efficiency.

Let us consider the problem of a large company which provides a fleet of trucks (or other vehicles) for use by numerous clients on a daily basis. The company must make long-range decisions to determine how large a fleet to maintain and where to locate it, which will require the establishment of one or more depot sites. Given these, the company must determine, on a periodic basis, the most efficient means of assigning its fleet to its clients depending upon the demand and limitations (constraints) imposed on the dispatcher. This is a combinatorial problem and is quite complex to solve. Besides, the complexity grows exponentially with problem size. It is this type of problem that we are addressing in this study.

Operations research literature refers to all three of the pre-
dominant problems of fleet management mentioned above, i.e., depot sitting, vehicle assignment, and fleet size determination. The initial operational problem of a new firm would be the selection of an optional number of vehicle depots at the best set of locations or sites. A similar decision process is required when an established firm desires to add a new depot or consolidate existing ones. A detailed discussion of this problem is given by Eilon, et al. [11], so it will not be further considered here. Similarly, fleet size determination is beyond the scope of this research. Once the appropriate vehicle assignment has been established, algorithms are available for generating the minimum fleet size required to satisfy the fixed schedule. Fleet size determination problems are discussed in several places [9, 16, and 27]. This research will address the problem of vehicle assignment.

Vehicle assignment is one of the common problems of operating transportation companies, and is the subject of this study. For the sake of clarity we will elaborate on certain key terms used in the study. First, the vehicle is that entity which is dispatched to satisfy the requirements of a client. A depot is some location where we can store and maintain our vehicles when they are not in use. And finally, a job is the location where the vehicle is utilized by a client, without any restrictions or controls imposed by the fleet owners. The environment with which we shall be dealing consists of a fleet manager who must allocate his vehicles to the jobs. He could choose to meet all of the job demands directly from the nearest
depot and return the vehicle directly to that depot upon completion of the job. This does not seem very efficient since some vehicles may be able to complete one job and proceed directly to another with some cost savings over the return-to-depot plan.

The motivation for this research comes from the case study by Gavish and Schweitzer [14] of a trucking company which had a vehicle assignment problem. The algorithm set forth in this paper is capable of handling a more complex and more realistic version of the problem addressed by them.

**Literature Review**

Throughout the literature on vehicle assignment problems various descriptors are used to title the vehicle assignment problem. These include vehicle dispatching, vehicle scheduling, vehicle assignment, delivery, transportation, and sequencing. There is no significant difference in meaning intended by these different titles. The primary distinction in vehicle assignment problems is between those that permit only one depot to be considered and those that consider several depots. We shall refer to these respectively as central-depot and multiple-depot problems. Secondary classification of the literature can be accomplished in terms of dynamic vs. static systems and stochastic vs. deterministic demands. The majority of the literature was found to be static and deterministic. Only Tillman [37] considers the stochastic demand conditions and only Gavish and Schweitzer [14] postulate a truly dynamic model.

Dantzig was the earliest referenced author to formulate what
we will term the central-depot vehicle assignment problem. It is interesting to trace his work in this area. First [8] he addressed a typical transportation problem of shipping a product from \( m \) sources to \( n \) destinations. Then Dantzig and Fulkerson [9] considered a fleet size determination problem, and in 1959 Dantzig and Ramser [10] formulated the first classical central-depot vehicle assignment problem.

Minor variations of this problem have been considered, resulting in additional solution difficulties. These variations are as numerous as the articles in this field and are not of sufficient importance to review in detail. The core central-depot problem can be stated as follows:

\[ N \text{ customers with known locations and demands are to be supplied from a single depot by vehicles of known capacity. The problem is to select the best routes subject to:} \]

(i) meeting all demands
(ii) not exceeding vehicle capacity
(iii) a maximum time limit for each route
(iv) a time interval within which a particular demand must be met

From 1962 to 1972, several authors presented solution procedures for the central-depot vehicle assignment problem and its variations. The resultant solution methodology was diverse in that some methods yielded optimal results and some only near optimal, and in that some methods were highly analytical and some highly heuristic. We find that simulation was used in three papers [4, 30, and 34]; integer programming in only one [1]; branch and bound techniques in four [5, 11, 18, and 33]; marginal analysis in five [5, 6, 13, 23,
and 38]; the 3-optimal tour method in two [5 and 11]; linear programming in two [8 and 21]; heuristic programming in nine [6, 10, 13, 17, 18, 23, 28, 38, and 43]; dynamic programming in one [19]; network techniques in two [2 and 20]; and one independent algorithm [22].

The central-depot problem is a special case of the multiple-depot problem, but the solution techniques of the central-depot problems do not extend themselves to the more general problem. For this reason we will not elaborate on the problem statements and solution techniques found in the literature on the central-depot problem.

The next logical advance in the state of the art was the formulation and solution of the multiple-depot vehicle assignment problem. Szwarc was the first author to formulate this problem in 1967. His problem [36] assumes the vehicles return to a depot after completing a job. Szwarc's solution procedure was developed in a transportation network format.

Some logical extensions of Szwarc's work would be the addition of a trans-shipment capability and provision for using arrival intervals for the job demands. These extensions were made as summarized in Table 2, but various solution techniques were used and each had its own peculiarities. Two heuristic programming solutions to the multiple-depot vehicle assignment problem were found [37 and 47]. Neither of these is able to guarantee that an optimal solution will be found, and both neglect the time interval restriction for meeting the demands. Two of the most recent treatments of this problem [3 and 32] also neglect the time interval restriction and the requirement for a vehicle to return to its depot of origin upon completion.
of its route. This latter constraint was satisfied by the previously mentioned formulations. Additionally, a linear programming approach [32] ignores the trans-shipments between jobs which are handled by [3] an out-of-kilter network solution. In each of the above cases the problem was defined so as to suit a particular technique. The solution procedures can not be extended to the more general problem we are considering and therefore, we will not dwell on any of these problem statements or solution techniques.

The primary reference for this research effort was a paper by Gavish and Schweitzer [14]. Their formulation neglects the requirement that a vehicle return to its original depot, but they are the first authors to consider the time interval constraints in a multiple-depot vehicle assignment context. Their solution methodology was of the transportation network form. Another consideration introduced to this problem by them is the inclusion of operational (dynamic) constraints on the compatibility of two jobs. This is easily handled by defining a feasible set of job combinations.

To summarize, the major constraints considered in the multiple-depot vehicle assignment problem are:

(i) meet all demands from the depots,
(ii) do not exceed vehicle capacity,
(iii) do not exceed an implicit maximum time limit for each route,
(iv) perform each demand within a specified time interval,
(v) allow for the serial combination of jobs, and
(vi) take into consideration compatibility constraints.

The present research has incorporated all of the constraints
which other multiple-depot problems have considered, with the exception of the re-dispatch provision. In addition, we have added the constraint that each vehicle must return to the depot of origin. Thus, this model is a more versatile one. The decomposition approach we use as part of our procedure has helped substantially to reduce computation time. Our solution technique will provide a near optimal solution with reasonable efficiency.

An overall appreciation of the literature can be obtained by the analysis of Tables 1 and 2. Table 1 presents a classification of articles by problem area with a detailed sub-division of the central-depot assignment problem solution techniques. At a glance one can see the time frame and quantity of work done in a particular area. The vehicle assignment literature cited is comprehensive, however the depot siting and fleet size references are not. These have been included only to provide a broader view of the environment of vehicle assignment problems. Much additional literature is currently available in these two areas. Table 2 provides a detailed analysis of the multiple-depot assignment problem statements and solution techniques used in the referenced literature and this paper.

The intent of this research is to formulate a multiple-depot vehicle assignment problem which will incorporate the key aspects mentioned above that have been omitted from previous work. With the exception of the re-dispatch provision, a general formulation of the problem will be solved by a combination of linear programming and network techniques and a near optimal solution achieved.
Table 1. Literature Synopsis

### Part 1.

| Reference | 8 9 10 19 1 6 20 4 13 17 18 21 23 34 36 2 |
| Year Published | 51 54 59 62 64 64 64 67 67 67 67 67 67 67 67 68 |

**Depot Siting**

Fleet Size +

**Multiple-Depot** +

**Central-Depot**

- Linear Prog. +
- Heuristic Prog. + + + + + +
- Dynamic Prog. +
- Integer Prog. +
- Marginal Anal. + + + +
- Network Tech. +
- Simulation + +
- Branch & Bound +

### Part 2.

| Reference | 22 38 43 5 16 33 37 27 11 39 3 28 32 47 14 30 |
| Year Published | 68 68 68 69 69 69 69 70 71 71 72 72 72 72 73 UNK |

**Depot Siting** +

**Fleet Size** + +

**Multiple-Depot** + + + + + +

**Central-Depot**

- Other +
- Marginal Anal. + +
- Heuristic Prog. + +
- Branch & Bound + + + +
- 3-Optimal Tour + +
- Simulation +
Table 2. Comparison of Multiple-Depot Formulations

<table>
<thead>
<tr>
<th>Reference</th>
<th>36</th>
<th>37</th>
<th>39</th>
<th>3</th>
<th>32</th>
<th>47</th>
<th>14</th>
<th>Present Research</th>
</tr>
</thead>
<tbody>
<tr>
<td>Year Published</td>
<td>67</td>
<td>69</td>
<td>71</td>
<td>72</td>
<td>72</td>
<td>72</td>
<td>73</td>
<td>74</td>
</tr>
</tbody>
</table>

Problem Criteria

<table>
<thead>
<tr>
<th>Problem Criteria</th>
<th>68</th>
<th>136</th>
<th>183</th>
<th>100</th>
<th>100</th>
</tr>
</thead>
<tbody>
<tr>
<td>Multiple-Depots</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Multiple Demand Points</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Transhipment allowed</td>
<td>NO</td>
<td>yes</td>
<td>yes</td>
<td>NO</td>
<td>yes</td>
</tr>
<tr>
<td>Return to Origin</td>
<td>yes</td>
<td>yes</td>
<td>NO</td>
<td>NO</td>
<td>yes</td>
</tr>
<tr>
<td>Cost Minimization</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Meet all Demands</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Arrival Interval</td>
<td>NO</td>
<td>NO</td>
<td>NO</td>
<td>NO</td>
<td>NO</td>
</tr>
<tr>
<td>Re-dispatch Allowed</td>
<td>yes</td>
<td>NO</td>
<td>NO</td>
<td>yes</td>
<td>NO</td>
</tr>
<tr>
<td>Compatibility Constraints</td>
<td>NO</td>
<td>NO</td>
<td>NO</td>
<td>NO</td>
<td>NO</td>
</tr>
<tr>
<td>Number Demands per Day</td>
<td>68</td>
<td>136</td>
<td>183</td>
<td>100</td>
<td>100</td>
</tr>
</tbody>
</table>

Solution Techniques

<table>
<thead>
<tr>
<th>Solution Techniques</th>
<th>+</th>
<th>+</th>
<th>+</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear Programming</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Heuristic Programming</td>
<td>+</td>
<td>+</td>
<td></td>
</tr>
<tr>
<td>Network Techniques</td>
<td>+</td>
<td>+</td>
<td></td>
</tr>
</tbody>
</table>
Problem Statement and Mathematical Model

To discuss the problem addressed in this study in detail, consider a fixed number of a particular type vehicle assigned to each of several depots. This will specify the number of depots and the quantity of vehicles available at each depot. Now a group of, say, J clients will make known their demands. Each demand will include the number of vehicles required, the type and quantity of cargo to be carried, any special features required on the vehicles, the starting time and place and the job duration or a specified release time and place. The fleet manager must now determine the costs and times for traveling to, from, and between the job locations. He must also specify a safety margin to be added to the time for each route and the maximum time a vehicle will be allowed to wait if it arrives prior to the job starting time, so that the drivers will not be allowed excessive idle time. Typically, there may be 50 to 100 demands per day, as in the case study [14] upon which this paper is based.

The solution of this problem involves the serial combination of jobs subject to the following nine constraints.

(i) The number of vehicles departing or returning to a depot cannot exceed the number assigned to that depot.
(ii) Vehicles must return to their depot of origin.
(iii) A vehicle may be dispatched only once. (By checking to see if an early returning vehicle can accept the entire route of a late starting vehicle, this restriction can be relaxed.)
(iv) The number of vehicles of a particular type departing a job must equal the number of that type that arrived at the job.
(v) The number of vehicles arriving at a job must equal the number requested.
(vi) Vehicles cannot arrive late for a job.
(vii) Vehicles cannot arrive too early for a job.
Successive cargos must be compatible
Successive special feature requirements must be compatible.

The objective function will be defined by a cost function which represents the cost of empty vehicles traveling the appropriate routes. This problem will only implicitly consider delivery aspects such as vehicle capacity. We will assume that the activity at a demand location is determined by the user and is therefore an uncontrollable factor for the fleet manager. Therefore, minimizing the costs associated with deadheading of the empty vehicles will minimize the controllable operating costs of the fleet.

In order to develop a solution technique for this multiple-depot vehicle assignment problem, we need to develop the mathematical model for the problem. A summary of the notation is presented in Table 3 for reference throughout the remainder of this paper.

Let us consider a situation with D depots, each having $S_i$, $i = 1, ..., D$ vehicles available for dispatch to J job sites, $D + 1$ to $D + J$, with demand requirements $M_j$, $j = D + 1, ..., D + J$. For convenience in notation we will let $I = D + 1$ and $K = D + J$. Additionally, let us adopt the superscript, $k$, where $k = 1, ..., D$ to represent the depot from which a particular set of vehicles originated. We shall refer to this as the kth type of vehicle but keep in mind that it only identifies the source of the vehicle and does not imply any physical differences between the vehicles.

Our decision variables will be $X^k_{ij}$ which will represent the number of vehicles, originating from the kth depot, to be sent from...
Table 3. Notation

<table>
<thead>
<tr>
<th>Notation</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C_{ij}$</td>
<td>The cost of traveling from depot i to job origin j where $i = 1, \ldots, D$ and $j = 1, \ldots, K$</td>
</tr>
<tr>
<td>$C_{ij}$</td>
<td>The cost of traveling from job i to job j where $i = 1, \ldots, K$ and $j = 1, \ldots, K$</td>
</tr>
<tr>
<td>$C_{ji}$</td>
<td>The cost of traveling from job release j to depot i where $j = 1, \ldots, K$ and $i = 1, \ldots, D$</td>
</tr>
<tr>
<td>$M_j$</td>
<td>The number of vehicles required at job j where $j = 1, \ldots, K$</td>
</tr>
<tr>
<td>$S_k^i$</td>
<td>The number of vehicles of type k assigned to depot i where $k = 1, \ldots, D$ and $i = 1, \ldots, D$ ($S_k^i = S_k$ if $k = 1$, 0 otherwise)</td>
</tr>
<tr>
<td>$J$</td>
<td>The total number of job demands</td>
</tr>
<tr>
<td>$D$</td>
<td>The total number of depots, also vehicle types</td>
</tr>
<tr>
<td>$I$</td>
<td>Equals D plus 1</td>
</tr>
<tr>
<td>$K$</td>
<td>Equals D plus J</td>
</tr>
<tr>
<td>$X_{kj}$</td>
<td>The number of vehicles of type k to be sent from location i to j (the decision variable) where $k = 1, \ldots, D$, $i = 1, \ldots, K$, and $j = 1, \ldots, K$</td>
</tr>
<tr>
<td>$\Omega$</td>
<td>The set of null variables</td>
</tr>
<tr>
<td>$t_{oj}$</td>
<td>The origination time for job location j where $j = 1, \ldots, K$</td>
</tr>
<tr>
<td>$t_{rj}$</td>
<td>The release time for job location j where $j = 1, \ldots, K$</td>
</tr>
<tr>
<td>$t_{ij}$</td>
<td>The travel time from job location i to j where $i = 1, \ldots, K$ and $j = 1, \ldots, K$</td>
</tr>
<tr>
<td>$a_{ij}$</td>
<td>The time safety margin for travel from job location i to j where $i = 1, \ldots, K$ and $j = 1, \ldots, K$</td>
</tr>
<tr>
<td>$w_j$</td>
<td>The maximum waiting time allowed prior to $t_{oj}$ at job location j where $j = 1, \ldots, K$</td>
</tr>
<tr>
<td>$g_{ij}$</td>
<td>The special feature compatibility index from job location i to j ($g_{ij} = 1$ if cargos are compatible, 0 otherwise) where $i = 1, \ldots, K$ and $j = 1, \ldots, K$</td>
</tr>
<tr>
<td>$f_{ij}$</td>
<td>The special feature compatibility index from job location i to j ($f_{ij} = 1$ if features are compatible, 0 otherwise) where $i = 1, \ldots, K$ and $j = 1, \ldots, K$</td>
</tr>
</tbody>
</table>
location \( i \) to location \( j \). Thus we need to minimize the costs of sending empty vehicles of all types from any depot to any job plus the cost of trans-shipping empty vehicles from any job to any other job and finally the costs of returning empty vehicles of the \( k \)th type to the correct originating depot from any job site. These three costs can be respectively expressed mathematically as our objective function:

\[
\text{Min } f(X) = \sum_{i=1}^{D} \sum_{j=1}^{K} C_{ij} X_{ij} + \sum_{i=1}^{K} \sum_{j=1}^{D} C_{ij} X_{ij} + \sum_{j=1}^{K} \sum_{i=1}^{D} C_{ji} X_{ji}
\]

where \( C_{ij} \) is the cost of sending one vehicle from depot \( i \) to job origin \( j \); \( C_{ij} \) is the cost of sending one vehicle from job \( i \) to job \( j \); and \( C_{ji} \) is the cost of returning one vehicle from job \( j \) to depot \( i \).

Now that we have derived our objective function, we must turn our attention to the constraints of the problem. First we must prohibit the dispatch of more vehicles than are available at a particular depot. Clearly there will be no vehicles at depot \( i \) of any type other than \( k = i \), since \( k \) is an index of the depot of origin. So let us define the vehicles of type \( k \) available at depot \( i \) to be \( S_{i}^{k} \), where \( S_{i}^{k} = 0 \) if \( k \neq i \). Thus our constraint for the limitation on supply can be written as:

\[
\sum_{j=1}^{K} X_{ij}^{k} \leq S_{i}^{k} \quad \text{for all } i = 1, \ldots, D \text{ and } k = 1, \ldots, D
\]
Furthermore, we must be able to guarantee that all of the vehicles that leave a given depot return to that depot at the end of the day.

\[ \sum_{j=1}^{K} x_{ij}^k = \sum_{j=1}^{K} x_{ij}^k \quad \text{for all } i = 1, \ldots, D \text{ and } k = 1, \ldots, D \quad (3) \]

Equation (2) and (3) are the mathematical representation of conditions (i), (ii), and (iii) as they were described in our problem statement earlier. These are the constraints on the depot environment of our problem and in network theory terminology are the node balance equations for the depots. Now we must examine the job environment and derive the job node balance equations.

First, we will consider the problem of balancing the arrival and departure of the kth type vehicle at job j. These vehicles can arrive (in general) from any depot or be trans-shipped from some other job. Similarly, these vehicles will be sent out from job j to depots or other jobs. Hence we require four terms to express the overall balance of the kth type vehicle at job j.

\[ \sum_{i=1}^{D} x_{ij}^k + \sum_{i=1}^{K} x_{ij}^k - \sum_{i=1}^{D} x_{ji}^k - \sum_{i=1}^{K} x_{ji}^k = 0 \quad (4) \]

for all \( j = 1, \ldots, K \) and \( k = 1, \ldots, D \)

Also we must insure that the demand of each job site is satisfied, since we assumed that we had the capability to accomplish this and we do not want to disappoint our customers. Here we must consider the inflow of vehicles of all types from all depots and all other
jobs which must equal the number of vehicles demanded, i.e.

\[
\sum_{i=1}^{D} \sum_{k=1}^{D} x_{ij}^k + \sum_{i=1}^{K} \sum_{k=1}^{D} x_{ij}^k = M_j \quad \text{for all } j = 1, \ldots, K
\]  

Equations (4) and (5) are the mathematical formulation of conditions (iv) and (v), respectively of the problem statement. We also need to apply a nonnegativity constraint on our decision variables and require them to be integer valued since we are dealing with whole vehicles.

\[
\text{all } x_{ij}^k \geq 0, \text{ integer}
\]  

Equation (1) subject to the constraints of equations (2) through (6) represents the principle expression of the general problem. Our solution technique will focus on these equations, which describe the static aspects of our stated problem.

Following Gavish and Schweitzer [14] we will consider the dynamic aspects of the problem separately.

As defined in Table 3, we will set \( x_{ij}^k = 0 \) if \( x_{ij}^k \in \Omega \) i.e., the set \( \Omega \) defines the set of null variables. Now

(a) \( x_{ij}^k \in \Omega \) if \( t_i^r + t_{ij} + h_{ij} \geq t_j^o \) i.e., if the release time at node \( i \) plus the travel and safety time between \( i \) and \( j \) exceed the starting time at node \( j \), then \( x_{ij}^k \) is a null variable.
(b) $X_{ij}^k \in \Omega$ if $t_i^r + t_{ij} \leq t_j^o - w_j$ i.e., if the earliest (release time plus only travel time) the vehicle can arrive from node $i$ at node $j$ is more than the prescribed waiting time before the start of job $j$, then $X_{ij}^k$ is a null variable.

(c) $X_{ij}^k \in \Omega$ if $g_{ij} = 0$ i.e., if jobs $i$ and $j$ are not compatible by the nature of their cargo.

(d) $X_{ij}^k \in \Omega$ if $f_{ij} = 0$ i.e., if jobs $i$ and $j$ are not compatible due to special requirements.

The above specifications correspond to conditions (vi) through (ix) of the problem statement. For $X_{ij}^k \in \Omega$, we force $X_{ij}^k = 0$ by setting $c_{ij}^k = \infty$.

There is one other major simplification which can be made to reduce the complexity of our problem expression.

Lemma 1. For each $i = 1, \ldots, D$ and $j = 1, \ldots, K$, $X_{ij}^k = X_{ji}^k = 0$ when $k \neq i$ in any feasible solution to the problem.

Proof. There are no vehicles from other depots allowed at depot $i$, hence, the definition of $S_i^k = 0$ for $i \neq k$. Now from equation (2) we can see that $\sum_{j=1}^{K} x_{ij}^k \leq s_i^k = 0$ for $i \neq k$ and we know that the sum of nonnegative (from equation (6)) variables must be nonnegative and we get: $0 \geq \sum_{j=1}^{K} x_{ij}^k \leq 0$ for $i \neq k$, hence $\sum_{j=1}^{K} x_{ij}^k$, $i \neq k$ must equal zero. Then for $i \neq k$ we get equation (3) in the form $\sum_{j=1}^{K} x_{ji}^k = \sum_{j=1}^{K} x_{ij}^k = 0$ for $i \neq k$. We know that the only way a sum of nonnegative variables can be equal to zero is for all the variables to be individually equal to zero. Hence: $X_{ij}^k = X_{ji}^k = 0$. 
when \( i \neq k \).

By invoking the results of Lemma 1 we can eliminate several summations and match the \( k \)-type superscript with the \( i \)-depot subscript in certain of our equations. The problem can be restated as:

**Problem P**: Minimize

\[
\begin{align*}
f(X) &= \sum_{k=1}^{D} \sum_{j=I}^{K} c_{k} x_{kj}^{k} + \sum_{i=I}^{K} \sum_{j=I}^{K} c_{ij} X_{ij}^{k} \\
&\quad + \sum_{j=I}^{K} \sum_{k=1}^{D} c_{jk} x_{jk}^{k} 
\end{align*}
\]

Subject to:

\[
\begin{align*}
\sum_{j=I}^{K} x_{kj}^{k} &\leq s_{k}^{k} \quad \text{for all } k = 1, \ldots, D \quad (8) \\
\sum_{j=I}^{K} x_{jk}^{k} - \sum_{j=I}^{K} x_{kj}^{k} &= 0 \quad (9)
\end{align*}
\]

for all \( k = 1, \ldots, D \)

\[
\begin{align*}
x_{kj}^{k} + \sum_{i=I}^{K} x_{ij}^{k} - x_{jk}^{k} - \sum_{i=I}^{K} x_{ji}^{k} &= 0 \quad (10)
\end{align*}
\]

for all \( k = 1, \ldots, D \) and \( j = I, \ldots, K \)

\[
\begin{align*}
\sum_{k=1}^{D} x_{kj}^{k} + \sum_{i=I}^{K} \sum_{k=1}^{D} x_{ij}^{k} &= m_{j} \quad (11)
\end{align*}
\]

for all \( j = I, \ldots, K \)

and all \( x_{ij}^{k} \geq 0, \) integer \( (12) \)

where \( c_{ij}^{k} = \infty \) if \( x_{ij}^{k} \in \Omega \) for any \( k = 1, \ldots, D \) and any job combination \((i,j)\).

This formulation is clearly an integer programming problem but its size could still pose a significant computational difficulty even
if the integrality requirement is relaxed. The context of our problem is a daily determination of vehicle assignments for up to ten depots and up to 100 job demands. A simplex solution of this size problem would require 1,002,000 real variables plus 1120 slack and artificial variables and there would be 1120 constraint equations. Thus even a moderate size problem will require some ingenuity and innovation to make a linear programming solution technique feasible. The problem will lend itself well to three different solution techniques, as evidenced by the work of previous authors summarized in Table 2; however, this research will show that employing the Dantzig-Wolfe decomposition principle together with the use of a simple algorithm from network theory (namely, the shortest path algorithm) will yield an efficient algorithm.
CHAPTER II

SOLUTION PROCEDURE

The Algorithm

To solve the problem we will first identify the null variables by the four dynamic considerations discussed earlier. This leaves us with an integer programming problem. We will relax the integrality requirement and solve the resulting linear program by the Dantzig-Wolfe decomposition procedure. The subproblems are selected so that they are shortest-path problems for which efficient procedures for solving are available. The subproblem may have negative cycles and hence, the Dantzig, Blattner and Rao algorithm [49] is used. The specific details of the Dantzig-Wolfe procedure and the shortest-path algorithm are well known and are not discussed here. The solution thus obtained is modified to give an integer solution to problem P. A heuristic procedure is described for achieving this.

To solve the linear program, recall that the general matrix formulation of a decomposable problem is:

\[
\begin{align*}
\text{Min } f(X) &= \sum_{k=1}^{D} C_k X_k \\
\text{subject to: } \sum_{k=1}^{D} A_k X_k &= b_o \\
B_k X_k &= b_k \quad \text{for all } k = 1, \ldots, D \\
\text{all } X_k &\geq 0
\end{align*}
\]

(13) (14) (15) (16)
where D is the number of subproblems.

This is the form of our problem with equation (7) corresponding to (13), equations (8), (9), and (10) corresponding to the subproblem equation (15) for each k, and equation (11) corresponding to the coupling constraint of equation (14). In writing a detailed expression of the kth Dantzig-Wolfe subproblem we can omit the use of the k superscript since it is the same within each subproblem. We will also denote the modified cost coefficients by \( \hat{C}_{kj}^o \), \( \hat{C}_{ij} \), and \( \hat{C}_{jk}^r \) as obtained by the Dantzig-Wolfe decomposition principle. Thus we get:

Problem \( P_k \):

\[
\text{Min } f_k(X) = \sum_{j=1}^{K} \hat{C}_{kj} x_{kj} + \sum_{i=1}^{K} \sum_{j=1}^{K} \hat{C}_{ij} x_{ij} + \sum_{j=1}^{K} \hat{C}_{jk}^r x_{jk} \tag{17}
\]

subject to:

\[
\sum_{j=1}^{K} x_{kj} \leq S_k \tag{18}
\]

\[
\sum_{j=1}^{K} x_{jk} - \sum_{j=1}^{K} x_{kj} = 0 \tag{19}
\]

\[
x_{kj} + \sum_{i=1}^{K} x_{ij} - x_{jk} - \sum_{i=1}^{K} x_{ji} = 0 \tag{20}
\]

for all \( j = 1, \ldots, K \)

all \( x_{ij} > 0 \) \( \tag{21} \)

Each such subproblem has \( (J + 2) \) equations and \( (D^2 + 2DJ) \) variables. Although we could solve each subproblem by a standard linear programming code, there is a special structure to the subproblem for-
mulation which can be exploited to reduce the solution time for each subproblem. For this purpose consider problem $P_k - 1$ below (where $C_{kk} = \infty$) which is the problem of finding the shortest path from depot $k$ through the trans-shipment network and back to depot $k$. We can easily identify this shortest path by using the proposed shortest path algorithm [49]. Theorem 1 below shows how the solution to $P_k$ can be obtained from the solution to $P_k - 1$, $k = 1, \ldots, D$. Note that equation (24) is really a combination of equations (18) and (19).

Problem $P_k - 1$:

\[
\min f_k(X) = \sum_{j=1}^{K} C_{kj} X_{kj} + \sum_{i=1}^{K} \sum_{j=1}^{K} C_{ij} X_{ij} + \sum_{j=1}^{K} C_{jk} X_{jk}
\]

subject to:

\[
\sum_{j=1}^{K} X_{kj} = 1
\]

\[
X_{kj} + \sum_{i=1}^{K} X_{ij} - X_{jk} - \sum_{i=1}^{K} X_{ji} = 0
\]

for $j = k$ and $j = I, \ldots, K$

\[
\text{all } X_{ij} \geq 0
\]

The objective function coefficients in equation (22) may be such that there is a negative cycle in the network of problem $P_k - 1$. The Dantzig, Blattner, and Rao algorithm [49] can detect such a negative cycle, if one exists.

Theorem 1. Let $\hat{X}$ be a solution to problem $P_k - 1$ with $\hat{X}_{kp} = \hat{X}_{qk} = 1$, i.e., arcs $(k,p)$ and $(q,k)$ are in the shortest path. Then:

(i) If $f_k(\hat{X}) \geq 0$, then: $X^* = 0$ is the optimal solution to
problem $P_k$.

(ii) If $f_k(\hat{X}) < 0$ and finite, with $(k, p, i_1, i_2, \ldots, i_m, q, k)$ as the shortest path, then: the optimal solution to problem $P_k$ is given by $X^* = X^*_{kp} = \ldots = X^*_{imp} = X^*_{qk} = S_k$ and all other $X^*_{ij} = 0$.

(iii) If $f_k(\hat{X}) < 0$ and infinite, then let $i_{k_1}, i_{k_2}, \ldots, i_{k_m}, i_{k_1}$ be the cycle with negative length and $i_{k_j} \neq k$, $j = 1, \ldots, m$, then: $X^*_{k_1k_2} = X^*_{k_2k_3} = \ldots = X^*_{k_mk_1} = 1$ with all other $X^*_{ij} = 0$, is an extreme ray of problem $P_k$.

**Proof.**

(i) Clearly $X = 0$ is feasible to $P_k$. Suppose $X^*$ be a solution to $P_k$ with $f_k(X^*) < 0$. This implies there exists a negative cycle $i_1, i_2, \ldots, i_m, i_1$. Then $\hat{X}_{i_1i_2} = \ldots = \hat{X}_{i_mi_1} = 1$ and all other $\hat{X}_{ij} = 0$ is a feasible solution to $P_{k-1}$ with $f_k(\hat{X}) < 0$, this contradicts the assumption that $\hat{X}$ solves $P_{k-1}$.

(ii) Suppose $X' \neq X^*$ is optimal to $P_k$, i.e., $f_k(X') < f_k(X^*)$.

Let $X'_{k_1i_1} = X'_{i_1i_2} = \ldots = X'_{i_mk} = S_k$. Then: $X''_{k_1i_1} = X''_{i_1i_2} = \ldots = X''_{i_mk} = 1$ is a feasible to $P_{k-1}$ with $f_k(X'') < f_k(\hat{X})$, a contradiction.

(iii) Substituting the solution $X^* \geq 0$ in equations (18), (19), and (20) we get:

$$\sum_{j=1}^{K} X^*_{kj} = 0$$
\[ \sum_{j=1}^{K} x_{jk}^* - \sum_{j=1}^{K} x_{kj}^* = 0 \]

\[ x_{kj}^* + \sum_{i=1}^{K} x_{ij}^* - x_{jk}^* - \sum_{i=1}^{K} x_{ji}^* = 0 \quad \text{for all } j = 1, \ldots, K \]

Hence, it is an extreme ray. This completes the proof.

It may be noted that case (iii) cannot arise under the assumptions of our problem, since \( t_j^0 \) and \( t_j^r \) are fixed. Hence from the four conditions resulting in null variables discussed earlier, either \( C_{jk} = \infty \) or \( C_{kj} = \infty \) for each \( j, k = 1, \ldots, K \). On the other hand, suppose \( t_j^0 \) is stated as an interval of time during which a job can start, i.e., the earliest starting time \( (t_j^E) \), the latest starting time \( (t_j^L) \), along with job duration time \( (d_j) \) is specified. In this situation case (iii) can occur.

It may be useful at this point to illustrate and discuss the three cases in Theorem 1. An example problem discussed later will contain each of the above cases. Consider iteration 1 of the example problem, the solution of \( P_2 - 1 \) is the shortest path shown in figure 1.

\[ \text{Figure 1. Illustration of case (i) of Theorem 1.} \]

We can send one vehicle along this shortest path at a positive cost of 7. Hence, the solution to problem \( P_2 \) is all \( X_{ij} = 0 \), an example of case (i).
In iteration 2 we find that $P_2 - 1$ has identified a negative cycle with $f_2(X) = -\infty$ as shown in case (iii). The cycle is $(3 - 4 - 3)$ which is diagrammed in Figure 2.

![Figure 2](image)

Figure 2. Illustration of case (iii) of Theorem 1.

Here the optimal solution to $P_2$ is unbounded. The extreme ray will have $X_{34} = X_{43} = 1$ and all other $X_{ij} = 0$.

Finally, case (ii) appears in iteration 3, where $P_2 - 1$ determines a shortest path with finite negative length, as shown in Figure 3 below.

![Figure 3](image)

Figure 3. Illustration of case (ii) of Theorem 1.

Here we find the solution to $P_2$ is $X_{24} = X_{42} = 6$ and all other $X_{ij} = 0$.

From the above discussion, we see that in any finite solution to $P_k$, the value of each variable is either 0 or $S_k$. Likewise, in the case of an unbounded solution, the extreme ray will have either 0 or 1 as components. In the Dantzig-Wolfe decomposition procedure,
each such solution is converted into a column of the restricted master problem, if the solution has not already been considered and if it will improve the objective function. From equation (11) it will be seen that such a column will have an entry, \( a_j \), in row \( j \), \( j = 1, ..., J \) as follows:

\[
a_j = \begin{cases} 
0 & \text{if case (i) of Theorem 1 arises.} \\
S_k & \text{if case (ii) of Theorem 1 arises and job } j \text{ is in the shortest path for depot } k. \\
1 & \text{if case (iii) of Theorem 1 arises and job } j \text{ is in the cycle.}
\end{cases}
\]

Solution of the restricted master problem will yield a vector \( \pi \), of dual variable values, which is used to modify the cost coefficients of the subproblems. Appendix B discusses how the \( \pi \) values can be calculated from the current tableau. The procedure terminates when no vector can be added to the restricted master problem which improves the objective function. At this point, the solution is in terms of \( \lambda^k_1 \), the convex combination of extreme points and positive linear combination of extreme rays. This is readily converted to the solution in terms of \( X^k_{1j} \).

Now using the matrix notation from the beginning of this chapter we can summarize and outline the step by step procedure of our algorithm.

(i) Solve each subproblem \( B_kX_k = b_k \) and obtain the shortest path or identify a negative cycle

(ii) Convert the subproblem solutions to columns of the restricted
master problem \[ \sum_{k=1}^{D} A_k x_i^k \lambda_i^k = b_i; \sum_{i=1}^{m} \lambda_i^k = 1 \]
for each \( k = 1, \ldots, D \) where the coefficient of \( \lambda_i^k \)
for the \( i \)th vector added is 1 for an extreme point
solution \( x_i^k \).

(iii) Solve the restricted master problem and obtain a vector
of dual variable values, \( \pi \).

(iv) Modify \( C_k \) by subtracting \( \pi A_k \) and resolve the subproblems.

(v) Stop when no new subproblem solutions can be added to
the restricted master problem.

(vi) Apply heuristic programming to restore integrality if
necessary.

**Integer Solution**

If a non-integer value appears in the optimal solution, the
same value will appear on all arcs of a cycle from some depot \( k \),
through a series of one or more jobs and back to \( k \). This is true
since, in our stated problem, case (iii) of Theorem 1 cannot arise
and the optimal solution is simply a convex combination of extreme
points, each representing one cycle from a given depot and back.

The integrality along a cycle can be enforced either by increas-
ing or decreasing the flow. If we reduce the flow, we will fail to
meet the demands of the jobs along the cycle but we require that all
demands be met in our problem statement. Initially we assumed that
we had sufficient availability to meet all demands. Hence, we can
increase the flow on all non-integer cycles to the next greater inte-
ger. This solution obviously can fail to be optimal since we have
increased the costs and will be allowing excess vehicles at some jobs.
Further Analysis of Solution

Now if $t_j^0$ is specified in terms of a time interval (along with a job duration time), case (iii) of Theorem 1 can occur. In this case we know $t_j^E$ and $t_j^L$, the earliest and latest times when the job can start and also the job duration $d_j$. And the solution of problem $P$ need not be optimal. To see this, note that constraint (11) of problem $P$ permits a demand, $M_j$, to be met only from vehicles arriving from a depot or from another job. In other words, it does not permit a vehicle to meet a unit of demand at $j$ and continue on to meet one or more units of demand at the same job site even though it can be done without violating the dynamic constraints. If this happens, the following heuristic procedure can be applied to reduce the value of the objective function.

(i) Let $i_1, i_2, ..., i_k, i_l$ denote a cycle in the solution with $X_{i_1i_2} = X_{i_2i_3} = ... = X_{i_ki_l} > 0$ for $j \in S$ $(j \neq i_2, ..., i_k)$ for some $k$. Let $d_i$ represent the job duration for meeting one unit of requirement at job $i$.

(ii) For each $X_{jil}^k$, $j \in S$, let $t^r$ be its actual release time obtained from the schedule given by the algorithm. Let

$$z_j^{(1)} = \left\lceil \frac{t_{i_2}^L - (t^r + t_{i_1i_2} + h_{i_1i_2})}{d_{i_1}} \right\rceil$$

where $[a]$ denotes the largest integer less than or equal to $a$.

(iii) Let $S_1 = \{j : j \in S$ and $z_j^{(1)} \geq 1\}$. Stop if $S_1 = \emptyset$ since this means we cannot reduce cycles without violating time
constraints. Otherwise, select a $q \in S_1$ and let $X_{k_1}^k = p$.

Now let $z_q^{(2)} = \text{Max} \{\lfloor K/p \rfloor, 1\}$ where $\lfloor K/p \rfloor$ is the largest integer $\leq K/p$. If $K > p$, $z_q^{(2)}$ measures the maximum number of cycles that the $p$ vehicles can perform. If $K \leq p$, $z_q^{(2)} = 1$ means $K$ of the $p$ vehicles can remove the cycle.

(iv) Let $z = \text{Min} (z_q^{(1)}, z_q^{(2)})$. Decrease $X_{i_1i_2}^{1}, \ldots, X_{i_{k-1}i_k}^{1}$ and $K$ by $(z \cdot p)$ if $K > p$ and by $(z \cdot K)$ if $K \leq p$.

An Example

The following example is presented for the purpose of clarifying the solution procedure and is not to be considered as a realistic problem. This example will demonstrate the use of the Dantzig-Wolfe decomposition algorithm and Dantzig, Blattner, and Rao's algorithm for the shortest path problem. It is expected that the reader is familiar with these procedures, and therefore their details will be shown but not extensively treated. The example is designed so that case (iii) of Theorem 1 does arise. For obtaining this case, we had to express $t_{ij}^0$ as an interval, such that any starting time for job $j$ was acceptable.

Consider a situation with two depots (#1 and #2) and two job locations (#3 and #4). The cost matrix for this case is:

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>12</td>
<td>15</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>7</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>6</td>
<td>3</td>
<td>∞</td>
<td></td>
</tr>
</tbody>
</table>
There are 15 and 6 vehicles available at depots 1 and 2 respectively. Job 3 requires 9 vehicles and job 4 requires 8. The situation is such that there are no compatibility restrictions and we are allowed to accomplish jobs 3 and 4 in either order without violating any time restrictions. This leads to the following mathematical statement of the problem according to our model in equations (7) through (12).

\[
\begin{align*}
\text{Min } f(X) &= 12x_{13}^1 + 15x_{14}^1 + 5x_{23}^2 + 2x_{24}^2 + 7x_{34}^1 + 7x_{34}^2 \\
&+ 3x_{43}^1 + 3x_{43}^2 + 17x_{31}^1 + 8x_{41}^1 + 2x_{32}^2 + 6x_{42}^2 \\
\text{subject to: } &\begin{align}
x_{13}^1 + x_{14}^1 &\leq 15 \\
x_{31}^1 + x_{41}^1 - x_{13}^1 - x_{14}^1 &= 0 \\
x_{13}^1 + x_{43}^1 - x_{31}^1 - x_{34}^1 &= 0 \\
x_{14}^1 + x_{34}^1 - x_{41}^1 - x_{43}^1 &= 0 \\
x_{23}^2 + x_{24}^2 &\leq 6 \\
x_{32}^2 + x_{42}^2 - x_{23}^2 - x_{24}^2 &= 0 \\
x_{23}^2 + x_{43}^2 - x_{32}^2 - x_{34}^2 &= 0 \\
x_{24}^2 + x_{34}^2 - x_{42}^2 - x_{43}^2 &= 0 \\
x_{13}^1 + x_{23}^2 + x_{43}^1 + x_{43}^2 &= 9 \\
x_{14}^1 + x_{24}^2 + x_{34}^1 + x_{34}^2 &= 8
\end{align}
\end{align*}
\]
all $X_{ij}^k \geq 0$

The equations denoted by (26) and (27) correspond to subproblems one and two, respectively. Our first step is then to solve the subproblems by finding the shortest path through the cost matrix given above. Using Dantzig, Blattner, and Rao's algorithm (which can be visually verified for this trivial example) we get the shortest paths of 1 to 4 to 1 and 2 to 3 to 2. Since there is a positive cost associated with both of these routes we will send 0 vehicles along each route. This will add a zero vector to the restricted master problem corresponding to constraints (28) above. Since this is an extreme point solution there will be a 1 in the appropriate linear combination constraint equations. Here the coefficients in the objective function for the artificial variables were taken as 9999. Omitting the artificial variables for simplicity the restricted master problem will appear as follows.

Iteration 1:

<table>
<thead>
<tr>
<th>ROW</th>
<th>VARIABLES</th>
<th>RHS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$\alpha_1$</td>
<td>$\beta_1$</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

The resulting dual variable values are 9999 and 9999 corre-
sponding to the initial basic artificial variables of row 1 and 2, which relate to jobs 3 and 4, respectively. Now using these dual variable values we can modify the original cost matrix to obtain:

\[
\begin{array}{cccc}
1 & 2 & 3 & 4 \\
1 & \infty & \infty & -9987 & -9984 \\
2 & \infty & \infty & -9994 & -9997 \\
3 & 17 & 2 & \infty & -9992 \\
4 & 8 & 6 & -9996 & \infty \\
\end{array}
\]

Notice that a negative cycle has now been created between jobs 3 and 4. This will be detected by the shortest path algorithm and arises in both subproblems. Thus our solutions are 1 to 4 to 3 to 4 and 2 to 3 to 4 to 3. These negative cycles imply unbounded subproblem solutions and the cycles are extreme rays of the solution set. In this case, following the Dantzig-Wolfe decomposition procedure we must convert the extreme rays to columns of the restricted master problem without including the column as part of the linear combination constraints. The columns generated are:

\[
\begin{array}{c}
10 \\
1 \\
1 \\
0 \\
0 \\
\end{array}
\]  
\[
\begin{array}{c}
10 \\
1 \\
1 \\
0 \\
0 \\
\end{array}
\]

Adding the equivalent of these we get the following initial simplex
tableau.

Iteration 2:

<table>
<thead>
<tr>
<th>ROW</th>
<th>VARIABLES</th>
<th>RHS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>( \alpha_1 )</td>
<td>( \beta_1 )</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

This iteration yields dual variable values of \(-9989\) and \(9999\), which in turn give an adjusted cost matrix of:

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(\infty)</td>
<td>(\infty)</td>
<td>10001</td>
<td>(-9984)</td>
</tr>
<tr>
<td>2</td>
<td>(\infty)</td>
<td>(\infty)</td>
<td>9994</td>
<td>(-9997)</td>
</tr>
<tr>
<td>3</td>
<td>17</td>
<td>2</td>
<td>(\infty)</td>
<td>(-9992)</td>
</tr>
<tr>
<td>4</td>
<td>8</td>
<td>6</td>
<td>9992</td>
<td>(\infty)</td>
</tr>
</tbody>
</table>

The resulting shortest path solutions are 1 to 4 to 1 and 2 to 4 to 2 with finite negative objective function values. These routes convert to restricted master problem columns of:

<table>
<thead>
<tr>
<th>345</th>
<th>48</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>15</td>
<td>6</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
Neither of these have been previously considered so we will add both to the restricted master problem and get:

**Iteration 3:**

<table>
<thead>
<tr>
<th>ROW</th>
<th>VARIABLES</th>
<th>RHS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$a_1$ $\beta_1$ $\gamma$ $a_2$ $\beta_2$</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0 0 0 10 345 48</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0 0 0 1 0 0</td>
<td>9</td>
</tr>
<tr>
<td>2</td>
<td>0 0 0 1 15 6</td>
<td>8</td>
</tr>
<tr>
<td>3</td>
<td>1 0 0 1 0</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>0 1 0 0 1</td>
<td>1</td>
</tr>
</tbody>
</table>

Here the dual variable values are 9999 and -9989 which give the adjusted cost matrix:

<table>
<thead>
<tr>
<th></th>
<th>1 2 3 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$\infty$ $\infty$ -9987 10004</td>
</tr>
<tr>
<td>2</td>
<td>$\infty$ $\infty$ -9994 9991</td>
</tr>
<tr>
<td>3</td>
<td>17 2 $\infty$ 9996</td>
</tr>
<tr>
<td>4</td>
<td>8 6 -9996 $\infty$</td>
</tr>
</tbody>
</table>

This time we get shortest paths of 1 to 3 to 1 and 2 to 3 to 2 which generate the following columns:

<table>
<thead>
<tr>
<th></th>
<th>435</th>
<th>42</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>
Both columns are admissible and we get the following restricted master tableau.

**Iteration 4:**

<table>
<thead>
<tr>
<th>ROW</th>
<th>VARIABLES</th>
<th>RHS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$\alpha_1$</td>
<td>$\beta_1$</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

This time we obtain dual variable values of 7 and 3 and the adjusted cost matrix is:

\[
\begin{array}{cccc}
1 & 2 & 3 & 4 \\
\hline
1 & \infty & \infty & 5 & 12 \\
2 & \infty & \infty & -2 & -1 \\
3 & 17 & 2 & \infty & 4 \\
4 & 8 & 6 & -4 & \infty \\
\end{array}
\]

Here we get shortest paths of 1 to 3 to 4 to 1 and 2 to 4 to 3 to 2. The solution to subproblem 1 has a positive cost associated with it so this would generate a zero vector which we have already considered. Thus we need only add the column

\[
\begin{array}{c}
42 \\
6 \\
6 \\
0 \\
1 \\
\end{array}
\]
Adding this column we get:

**Iteration 5:**

<table>
<thead>
<tr>
<th>ROW</th>
<th>VARIABLES</th>
<th>RHS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$\alpha_1$</td>
<td>$\beta_1$</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

This results in dual variable values of 10 and 0 and an adjusted cost matrix of:

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>2</td>
<td>15</td>
</tr>
<tr>
<td>2</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>-5</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>17</td>
<td>2</td>
<td>$\infty$</td>
<td>7</td>
</tr>
<tr>
<td>4</td>
<td>8</td>
<td>6</td>
<td>-7</td>
<td>$\infty$</td>
</tr>
</tbody>
</table>

and the shortest path of 1 to 3 to 4 to 1 has a positive cost and 2 to 3 to 2 has already been considered.

The pertinent information from the last restricted master problem is

$$\beta_3 = \frac{1}{6}, \quad \beta_4 = \frac{5}{6}, \quad \gamma = 3$$
This tells us to use \((1/6 \cdot 6) = 1\) vehicle along route 2 to 3 to 2 and similarly 5 vehicles along route 2 to 4 to 3 to 2 and 3 vehicles along cycle 3 to 4 to 3.

Now, we shall examine whether the solution can be improved as discussed earlier. The reduction of cycling is dependent on the operational constraints so let us consider two cases:

**Case 1.** Suppose \(t_{34} = t_{43} = 20\) min. and \(h_{34} = h_{43} = 5\) min. and \(w_3 = w_4 = 10\) min. and \(t^E_3 = t^E_4 = 8\) am with \(t^L_3 = t^L_4 = 6\) pm with job durations at each job of one hour per load. Following our heuristic algorithm for breaking cycles, given earlier, we see that:

\[
z^{(1)}_2 = \left\lfloor \frac{6\text{ pm} - (9\text{ am} + 20\text{ min} + 5\text{ min})}{1\text{ hour}} \right\rfloor = 8
\]

and

\[
z^{(2)}_2 = \text{Max} \{ [3/5], 1 \} = 1
\]

Hence we can break the cycle by retaining 3 vehicles at job 4 for one additional job duration.

**Case 2.** Suppose we alter the job duration of job 4 to 2 hours and \(t^L_3 = 11\) am and leave the other parameters as stated in Case 1.

Here we calculate:

\[
z^{(1)}_2 = \left\lfloor \frac{11\text{ am} - (10\text{ am} + 20\text{ min} + 5\text{ min})}{2\text{ hours}} \right\rfloor = 0
\]

Hence, \(S_1 = \emptyset\) and the cycle cannot be broken.
This completes the solution of our small example problem. The computer solutions of other more complex and larger problems are summarized in Table 4.
CHAPTER III

COMPUTATIONAL RESULTS AND CONCLUSIONS

The algorithm was programmed in FORTRAN for solution on the Univac 1108 computer. No special attempt was made at optimizing the program or to reduce storage requirements. As will be seen, the computation time is very encouraging, but the program takes a large amount of storage.

The general flow diagram of the algorithm code is shown in Figure 4. A main program was used to control the basic data input and the sequencing of the solution procedure. Subroutines were used for modifying the cost matrix in response to the operational constraints, for adding columns to the restricted master problem, for solving the restricted master problem by linear programming, and for solving the subproblems by a shortest path method.

First the main program reads in the number of jobs and depots and initializes some defining parameters. Then the unaltered cost matrix is read from data cards. If any modifications are required to satisfy operational constraints then subroutine MODIFY is called to accomplish them. Variable initialization is also accomplished at this stage. Next, we set up the penalized cost matrix for the subproblems and solve each subproblem by calling subroutine SPATH. The shortest path solutions are saved and a vector generated for possible inclusion in the restricted master problem. The initial tableau of
Start

MODIFY: solve dynamic constraints and identify null variables

Set up subproblems

SPATH: obtain shortest path solutions

Interpret subproblem solutions

Are any columns to be added to RMP?

No

STOP

ADD: place new solution in RMP tableau

LINEAR: Solve RMP

Calculate adjusted cost matrix for next iteration

Figure 4. Flow Diagram of the Algorithm (for continuous solutions)
the restricted master problem is saved at each iteration and new columns added if appropriate by subroutine ADD. Each column of the restricted master tableau is matched with the saved shortest path which generated the column for ease in calculating the final solution. The restricted master problem is then solved by calling subroutine LINEAR. This provides a new set of dual values which are used to adjust the cost matrix for the subproblems. When no new columns can be added to the restricted master problem, the algorithm terminates and the main program calculates and prints the optimal solution.

Subroutine ADD takes a previously selected subproblem solution vector and converts it to a column of the restricted master problem to include the value of the linear combination component.

Subroutine MODIFY, if called, reads in the travel, waiting, and safety times and the compatibility conditions. The subroutine then checks to see if any of the operational constraints are violated and if so, causes the appropriate modification of the cost matrix to prevent the undesired combination.

The core of subroutine LINEAR is from the LINEAR-B program of Dr. Ronald L. Rardin (Georgia Institute of Technology). The subroutine uses a modified simplex method to solve a linear problem. The original program has been adjusted to perform minimization and to be compatible with the formats of this particular problem. New dual variable values are determined within the subroutine at each iteration and are transferred by a COMMON statement.

Subroutine SPATH finds the shortest path through the input
modified cost matrix. Using the methods and equations of the selected shortest path algorithm [49], negative cycles can be detected. A negative cycle implies an unbounded solution to the subproblem. The output of the subroutine is the shortest path (including any negative cycle found) and the total cost for one vehicle traveling that path.

All of the subroutines and the main program use basic FORTRAN techniques and are easily followed step by step. No special computer science techniques were utilized; nor are any required to work with this program.

The maximum size problem that the Univac 1108 can handle by our program is a 3-depot 18-job problem. This is because the program required an excess of 46 K of memory for data storage and manipulation. Through packing of matrices and the use of auxiliary tapes, the sizes of the problems handled can be substantially increased, and 10-depot 100-job problems will be within the capacity of the current approach. It may be noted that Gavish and Schweitzer [14] were able to achieve a significant reduction in computer time by reprogramming from FORTRAN H to ASSEMBLAR. It is anticipated that the application of similar techniques to this algorithm would achieve comparable results.

Test Problems

The following test problems in Table 4 were selected as typical of the model derived in this thesis. The number of jobs was varied in steps of three from 3 to 18. We elected to use only three
depots for the test problems since this provided a depot-to-job ratio varying from 1:1 to 1:6.

The cost data for each problem was generated by use of a multiplicative congruential random number generator, with the same random number seed used for each problem set. More details are given in Appendix A. Certain assumptions were made in the generation subroutine to make the formulation realistic.

First, we assumed that each depot had an equal density of jobs about it and that the vehicle availability at each depot exceeded the total demand of the jobs within that depot's area by 20 percent. Secondly, we assumed that the depot and job numbers provided a general measure of distance which could be used to realistically adjust the costs for the spread of the depots and jobs. Finally, we provided for an increase of return-to-depot costs ten percent of the time to account for overtime or other similar costs.

As stated in the example problem of Chapter II, the operational constraint aspects of this problem did not significantly affect the solution procedure. For this reason we have omitted the implementation of these constraints in our test problems. As discussed earlier, our problem has either $C_{ij} = \infty$ or $C_{ji} = \infty$, which was insured in our data generation method.

Table 4 lists the details of the problems solved and computational results. The column R. N. Seed gives the random number seed for the purpose of reproducing these problems if necessary.
Table 4. Test Problem Solutions

<table>
<thead>
<tr>
<th>Problem</th>
<th>No. of Depots</th>
<th>No. of Jobs</th>
<th>R. N. Seed</th>
<th>Total No. Cols. in RMP</th>
<th>No. Basic Cols. in Opt. Soln.</th>
<th>Max. Job/Route</th>
<th>Execution Time (Sec.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
<td>3</td>
<td>9573</td>
<td>11</td>
<td>3</td>
<td>1</td>
<td>.19</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>6</td>
<td>9573</td>
<td>24</td>
<td>6</td>
<td>2</td>
<td>.77</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>9</td>
<td>9573</td>
<td>24</td>
<td>9</td>
<td>3</td>
<td>1.55</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>12</td>
<td>9573</td>
<td>76</td>
<td>11</td>
<td>4</td>
<td>9.75</td>
</tr>
<tr>
<td>5</td>
<td>3</td>
<td>15</td>
<td>9573</td>
<td>79</td>
<td>13</td>
<td>3</td>
<td>14.10</td>
</tr>
<tr>
<td>6</td>
<td>3</td>
<td>18</td>
<td>9573</td>
<td>89</td>
<td>10</td>
<td>4</td>
<td>24.34</td>
</tr>
<tr>
<td>7</td>
<td>3</td>
<td>3</td>
<td>8537</td>
<td>14</td>
<td>3</td>
<td>1</td>
<td>.31</td>
</tr>
<tr>
<td>8</td>
<td>3</td>
<td>6</td>
<td>8537</td>
<td>29</td>
<td>4</td>
<td>3</td>
<td>1.05</td>
</tr>
<tr>
<td>9</td>
<td>3</td>
<td>9</td>
<td>8537</td>
<td>23</td>
<td>7</td>
<td>2</td>
<td>1.14</td>
</tr>
<tr>
<td>10</td>
<td>3</td>
<td>12</td>
<td>8537</td>
<td>106</td>
<td>9</td>
<td>4</td>
<td>27.05</td>
</tr>
<tr>
<td>11</td>
<td>3</td>
<td>15</td>
<td>8537</td>
<td>46</td>
<td>12</td>
<td>3</td>
<td>6.74</td>
</tr>
<tr>
<td>12</td>
<td>3</td>
<td>18</td>
<td>8537</td>
<td>101</td>
<td>15</td>
<td>3</td>
<td>32.16</td>
</tr>
<tr>
<td>13</td>
<td>3</td>
<td>3</td>
<td>9217</td>
<td>12</td>
<td>4</td>
<td>1</td>
<td>.25</td>
</tr>
<tr>
<td>14</td>
<td>3</td>
<td>6</td>
<td>9217</td>
<td>33</td>
<td>6</td>
<td>3</td>
<td>1.63</td>
</tr>
<tr>
<td>15</td>
<td>3</td>
<td>9</td>
<td>9217</td>
<td>34</td>
<td>9</td>
<td>5</td>
<td>2.28</td>
</tr>
<tr>
<td>16</td>
<td>3</td>
<td>12</td>
<td>9217</td>
<td>25</td>
<td>9</td>
<td>2</td>
<td>2.10</td>
</tr>
<tr>
<td>17</td>
<td>3</td>
<td>15</td>
<td>9217</td>
<td>40</td>
<td>12</td>
<td>3</td>
<td>5.80</td>
</tr>
<tr>
<td>18</td>
<td>3</td>
<td>18</td>
<td>9217</td>
<td>84</td>
<td>12</td>
<td>3</td>
<td>27.13</td>
</tr>
<tr>
<td>19</td>
<td>3</td>
<td>3</td>
<td>8991</td>
<td>12</td>
<td>3</td>
<td>1</td>
<td>.30</td>
</tr>
<tr>
<td>20</td>
<td>3</td>
<td>6</td>
<td>8991</td>
<td>87</td>
<td>6</td>
<td>2</td>
<td>6.69</td>
</tr>
<tr>
<td>21</td>
<td>3</td>
<td>9</td>
<td>8991</td>
<td>39</td>
<td>7</td>
<td>3</td>
<td>2.44</td>
</tr>
<tr>
<td>22</td>
<td>3</td>
<td>12</td>
<td>8991</td>
<td>37</td>
<td>8</td>
<td>3</td>
<td>4.07</td>
</tr>
<tr>
<td>23</td>
<td>3</td>
<td>15</td>
<td>8991</td>
<td>72</td>
<td>12</td>
<td>4</td>
<td>12.92</td>
</tr>
<tr>
<td>24</td>
<td>3</td>
<td>18</td>
<td>8991</td>
<td>38</td>
<td>15</td>
<td>2</td>
<td>5.93</td>
</tr>
</tbody>
</table>
Conclusions

Most of the published studies do not consider the requirement that a vehicle return to its depot of origin. By considering a rather general model with this added constraint, this study hopefully represents an advancement in this area of research. Also, through the use of the Dantzig-Wolfe decomposition procedure and a shortest path algorithm it is shown that reasonably sized problems can be solved effectively.

In applying the proposed solution technique, computer memory limitations were encountered. This forced us to work with relatively small problems, but the results of Table 4 show that the computation times are relatively small. As discussed earlier, more advanced programming methods should be able to solve larger problems.

It may be noted from Table 4 that problem 10 took substantially more time to solve than problems 4, 16, and 22 for the same number of depots and jobs. The total number of columns considered in the restricted master problem was also significantly higher in this case. The increased solution time probably came from the repeated execution of LINEAR or SPATH subroutines.

We discussed in some detail the implications of extending our model to incorporate the specification of a starting interval and job duration for job $j$, in lieu of the fixed starting time $t_j^0$. Other manipulations of the dynamic constraint parameters can be handled with less difficulty, e.g., any particular routing may be prohibited simply by setting the appropriate compatibility indices to zero.
Recommendations

This research has pointed to areas where further work is indicated. Effort expended on the following extensions would improve the applicability of this algorithm to the transportation field.

(i) How can we minimize the size of the fleet without increasing the overall costs?

(ii) Can the fleet be redistributed among the depots to further reduce the operating costs?

(iii) How can compatibility for a series of three or more jobs be expressed?

(iv) How can re-dispatch of a vehicle be allowed simultaneously with consideration of arrival intervals?

(v) What programming techniques can be applied to this FORTRAN program in order to reduce the core storage requirement?

(vi) Can the dynamic aspects of the problem be included in the analytical model and an optimal integer solution achieved?

These areas in no way imply inadequacy of the present research but are simply beyond the scope of this work.
APPENDIX A

Main Program User's Instructions

The program is written to accept punch cards in the following order and format:

(i) Number of depots, number of jobs (integers)
(ii) One card for each depot containing the depot to job costs in sequence separated by commas, followed by the availability at that depot (floating point)
(iii) One card for each job containing the job to depot and then the job to job costs all in sequence, followed by the demand for that job (floating point)
(iv) A one integer code, when set to zero, will prevent compatibility modifications. If set to an integer other than zero, MODIFY will be called to make the desired changes. (integer)

If subroutine MODIFY is to be used, additional data cards are required as follows:

(v) Number of incompatible entries (integer)
(vi) One card for each incompatible entry giving the pair of ordered job numbers which are not compatible separated by commas. (integers)
(vii) One card for each depot giving the travel time to each
job in sequence separated by commas (floating point)

(viii) One card for each job giving the travel time to each depot and then to each job in sequence separated by commas followed by the start and release times for that job (floating point)

(ix) One card for each depot giving the safety time to each job in sequence separated by commas (floating point)

(x) One card for each job giving the safety time to each depot and then to each job in sequence separated by commas followed by the maximum waiting time allowed at that job (floating point)

To clarify the use of the data cards the following set would be required to run the example problem of Chapter II.

2, 2
12., 15., 15.
5., 2., 6.
8., 6., 3., 9999., 8.
0

The output of the program states how many vehicles to send on a particular route. Thus the dispatcher has only to assign a certain number of his vehicles to the specified route. A sample output line would read, "Send 8. vehicles along route 1, 4, 3, 1," which means
that 8 vehicles are to be sent from depot 1 to job 4, then on to job 3 and finally back to depot 1.
MAIN PROGRAM FORTRAN

SOURCE CODE LISTING
FELCH-JHON-E*TRANS, MAIN

C  ****DATA READ-IN AND INITIALIZATION  ****
C
CO*WOV C(150), CSAVE(150), COST(25, 26), DUAL(45), LABEL(25, 150)
INVECT(150), R/P(45, 150), VECTOR(50, 150), D, I, J, K, L, M, JN, J0, NTOT, I Short(222), COSTM(25, 26), NMIT, CTOTAL, CYNEG
DIMENSION B(45, 153), IHSV(5, 30, 22)
INTEGER LABEL, D
READ(5, 10) D, J
FORMAT ( )
I=I+1
K=I+J
L=K+1
M=K+2
J=2*J+2
J=N=2*J+2
JL=J+1
J=J+1
N=2*J
J=2*J+1
DO 10 II=1, D
NVECT(II)=0
10 CONTINUE
N=I
DO 20 II=1, J
DUAL(II)=0.
20 CONTINUE
DO 25 II=1, D
COSTM(II,J)=9999.
25 CONTINUE
DO 30 II=1, D
READ(5, 10)(COSTM(II, J), J=I, L)
30 CONTINUE
DO 35 II=1, D
READ(5, 10)(COST(II, J), J=I, L)
35 CONTINUE
DO 40 II=1, D
READ(5, 10) NN
1(FINN,EQ., 0) GO TO B5
40 CONTINUE
CALL MODIFY
45 DO 40 II=1, K
COSTM(II, IJ)=COSTM(II, IJ)-DUAL(II)
40 CONTINUE
IF(II, EQ., IJ) COSTM(II, IJ)=9999.
45 CONTINUE
45 CONTINUE
50 C  ****SOLUTION OF SUBPROBLEMS  ****
C
DO 100 II=1, D
DO 110 II=I, K
IK=I-1
I=I-1
100 CONTINUE
COSTM(I, IJ)=COSTM(I, IJ)-DUAL(IK)
110 CONTINUE
C 100 CONTINUE
COSTM(I, IJ)=COSTM(I, IJ)-DUAL(IK)
57  COSTm(IJ, L) = COST(IJ, II)
58  COSTm(IJ, M) = 9999.
59  COSTm(IJ, L) = 9999.
60  CONTINUE
61  COSTm(L, L) = 9999.
62  COSTm(L, M) = 9999.
63  COSTm(L, L) = 9999.
64  JJ, 130 IJ = 1, JM
65  VECTOR(IJ, IJ) = 0.
66  CONTINUE
67  130 CONTINUE
68  140 CALL SPATH(IJ)
69  IK = J + 2
70  DO 145 IJ = 1, IK
71  ISV(IJ, НИТ, IJ) = ISHORT(IJ)
72  145 CONTINUE
73  DO 160 IJ = 1, K
74  IK = IJ - 3
75  IF(ISHORT(2, EQ, IJ, AND, CYNEG, GE, 0.)) VECTOR(IJ, K) = COST(IJ, L)
76  IK = J + 2
77  DO 150 IL = 3, IR
78  IO = ISHORT(IL)
79  IF(IO, EQ, IJ, OR, IO, EQ, 0.) GO TO 160
80  IF(IO, EQ, IJ) VECTOR(IJ, IL) = VECTOR(IJ, IK) + COST(IJ, L)
81  150 CONTINUE
82  160 CONTINUE
83  VECTOR(IJ, JM) = CTOTAL * COST(IJ, L)
84  180 CONTINUE
85  C*****SET UP OF INITIAL RESTRICTED MASTER PROBLEM
86
87  IF(NMrr, GT, I) GO TO 300
88  DO 220 II = 1, 150
89  DO 200 IJ = 1, I
90  LABEL(IJ, II) = 0.
91  RMP(IJ, II) = 0.
92  200 CONTINUE
93  DO 210 IJ = 1, K
94  RMP(IJ, II) = 0.
95  210 CONTINUE
96  220 CONTINUE
97  DO 230 II = 1, 150
98  CSAVE(II) = 0.
99  230 CONTINUE
100 NTOT = 1
101 DO 240 II = 1, 10
102 CALL ADD(II)
103 NTOT = NTOT + 1
104  240 CONTINUE
105 C*****ADDITION OF NEW SUBPROBLEM VECTORS TO THE RESTRICTED
106 C*****MASTER PROBLEM
107 C
108  300 CONTINUE
109  NTOT = 1
110 DO 310 II = 1, 10
111 NTUT = NTOT + NVEOT(II)
112  310 CONTINUE
I

NADD=0
DO 350 II=1,D
KTEST=VECTOR(II+JM)*100000
IF(KTEST.EQ.0) GO TO 350
NT=NTOT+1
DO 340 IJ=1,NT
IF(LABEL(IJ).EQ.0) GO TO 340
K=IJ
IF(RMP(IK,IJ).LE.0..AND.CYNEG.GE.0.) GO TO 340
IF(W4P(IK,IJ).GE.1.,AND.CYNEG.LT.0.) GO TO 340
DO 320 I<=1,J
ITST=KIP(IK,IJ)*100000
JTST=VECTOR(IK+IJ)*100000
IF(ITST.NE.JTST) GO TO 340
320 CONTINUE
IF(VECTOR(IK,IJ).GE.CSAVE(IJ)) GO TO 350
340 CONTINUE
CALL ADD(IK)
350 CONTINUE
IF(NADD.EQ.0.AND.N4IT.N.T) GO TO 2000
C****ADDITION OF ARTIFICIAL VARIABLES AND SOLUTION OF THE
C****RESTRICTED MASTER PROBLEM
C
IP=NTOT+K-1
DO 420 II=NTOT,IP
CSAVE(II)=9999.
IK=II-NTOT+1
DO 400 TJ=1,K
RMP(IK,IJ)=0.
400 CONTINUE
RMP(IK,IJ)=+1.
DO 410 IJ=1,I
410 CONTINUE
420 CONTINUE
IRHS=IP+1
DO 430 II=1,J
430 CONTINUE
DO 450 II=J+1,K
RMP(IK+II+1)+1.
450 CONTINUE
DO 500 II=1,K
DG 510 IJ=1,IRHS
R(IJ,IJ)=RMP(II+IJ)
510 CONTINUE
DO 520 II=1,IP
CI(IJ)=CSAVE(IJ)
520 CONTINUE
CALL LINEAR(K,IP,B,NMIT,45+150)
NMIT=NMIT+1
C

C MODIFICATION OF THE TRANSPORT COST MATRIX

DO 610 II=1,K
   DO 600 IK=1,K
   IJ=IK-1
   COSTM(II,IK)=COST(II,IK)-DUAL(IJ)
   IF(IJ.EQ.IK)COSTM(II,IK)=9999.
   600 CONTINUE
   610 CONTINUE
   GO TO 100

2000 ITFIN=MIT-1
   TO 2003 II=1,D
   GO 2001 IJ=1,NTOT
   IF(LABEL(II,IJ).NE.1) GO TO 2001
   X=COST(I100*VECTOR(ITFIN, IJ)
   IYFLA;3EL(1,1J)
   IZ=J+2
   WRITE(5,2002) XI(ISV(II, IY, IK)PIK=1eIZ)
   2002 FORMAT(14X,JOSENO ',1F5.2,' VEHICLES ALONG ROUTE '813,1(/),19X,17I
   2001 CONTINUE
   2003 CONTINUE
   WRITE(6,2004)
   2004 FORMAT(11v)
   ENJ

FELCH-JHON-E*TRANS.ADO

1 SUBROUTINE ADD(MJ)
2  COMPO: C(150),CSAVE(150),COST(25,26),DUAL(45),LABEL(25,150)*
3  INVECT(5),NP(45,150),VECTOR(50+150),1,I,J,K,L,M,JM,JH,NTOT,ISHORT(
4  222)COSTM(25,26),NMIT,CTOTAL,CYNEG
5  INTEGER LABEL,
6  DO 12 JM=1,D
7  LABEL(JM,NTOT)=0
8  12 CONTINUE
9  CSAVE(NTOT)=VECTOR(MJ,JM)
10  LABEL(JM,NTOT)=1
11  LABEL(I,NTOT)=NMIT
12  DO 42 JM=1,J
13  RXP(MJ,NTOT)=VECTOR(MJ,MJ)
14  42 CONTINUE
15  MN=JM+1
16  RXP(MN,NTOT)=1.
17  DO 52 KM=1,D
18  KPI=JM+1
19  IF(KM.EQ.MN,AND.CYNEG.GE.0.) GO TO 52
20  RXP(KM,NTOT)=0.
21  52 CONTINUE
22  NVECT(MJ)=NVECT(MJ)+1
23  RETURN
24 END
FELCH-JHON-E*TRANS,MODIFY

SUBROUTINE MODIFY

COMMON C(150),CSAVE(150),COST(25,26),DUAL(45),LABEL(25,150),
INVECT(5),MNP(45,150),VECTOR(50,150),D,I,J,K,L,KM,JM,JN,JTOT,JSHORT,
DIMENSION HANDW(25,26),ICOMP(25,25),TIME(25,27)

INTEGER LABEL,D

10 FORMAT()

DO 30 II=1,K
DO 30 II=1,K
ICOMP(II,II)=1

30 CONTINUE

READ(5,10) N
IF(N.E.0) GO TO 65
DO 60 II=1,N
READ(I,J) IK,IL
ICOMP(IK,IL)=0

60 CONTINUE

65 DO 90 II=1,K
DO 90 II=1,M
TIME(II,II)=0.

90 CONTINUE

DO 100 II=1,K
DO 100 II=1,M
TIME(II,II)=D.

100 CONTINUE

DO 110 II=1,L
READ(5,10)(TIME(II,IJ),I,J=II,K)

110 CONTINUE

DO 120 II=1,M
READ(5,10)(TIME(II,IJ),I,J=1,K)

120 CONTINUE

DO 130 II=1,L
READ(5,10)(HANDW(II,IJ),I,J=I,K)

130 CONTINUE

DO 140 II=1,L
READ(5,10)(HANDW(II,IJ),I,J=II,L)

140 CONTINUE

DO 150 II=I,K
IF(ICOMP(I,J).EQ.0) COST(I,J)=9999.
IF(TIME(I,I)+TIME(I,J)+TIME(I,J)+HANDW(I,J,J)-TIME(I,J,J).LE.0) COST(I,J)
=9999.
ELSE IF(TIME(I,K)+TIME(I,J)+HANDW(I,J,J)-TIME(I,J,J)+TIME(I,J,J).GE.0) COST(I,J)
=9999.

150 CONTINUE

210 CONTINUE

200 CONTINUE

RETURN

END
SUBROUTINE LINEAR(NOROW,NOCOL,A,N,JDIM,JDIM)

COMMON C(150),CSAVE(150),COST(25,26),DUAL(45),LABEL(25,150),
   IVEC(5),NP(45,150),VECTOR(50,150),O,J,K,L,M,N,NNOT,ISHORT(1),
   COSTM(25,26),TRANS,CYLINES

INTEGER LABEL

DIMENSION A(JDIM,JDIM),IBM(45),K8V(45),COLIN(45)

EPS=.000001
MAXCOL=150
MAXROW=45

IF(NOROW-ST.MAXROW,OR.NOCOL.GT.MAXCOL) GO TO 910
C(NORHS)=1,
DO 150 NR=1,NOROW
   IF(A(NR,NORHS),LT.0,) GO TO 920
   IF(IBM(NR).NE.0) GO TO 150
   CONTINUE

150 CONTINUE

C------GET INITIAL BASIS

NC=NORHS
DO 190 NR=1,NOROW
   NC=NC+1
   NOZER=0
   NOONE=0
   DO 180 NR=1,NOROW
      IF(A(NR,NC),GT.0,.AND,A(NR,NC).LT.-EPS) NOZER=NOZER+1
   180 CONTINUE
   IF(N00;1E.NE.1.0R.NOZER.NE.NOROW-1) GO TO 170
   IF(N00,NE.1.0R.NOZER,NE.NOROW-1) GO TO 170
   IF(IBM(NR),NE.0) GO TO 195
   NC=NC+1
   NOZER=0
   NOONE=0
   DO 193 NR=1,NOROW
      IF(IBM(NR),NE.0) GO TO 195
      NOZER=0
      NOONE=0
      WRITE(*,193) NR
   193 FORMAT('NO STARTING BASIC SOLN FOR ROW ',I5)
   195 CONTINUE
   IF(NO3Sw.EQ.1) GO TO 995

C------ELIMINATE BASIC VARIABLES FROM OBJECTIVE FUNCTION
   DO 197 NC=1,NORHS
   197 CONTINUE
   IF(NO3Sw.EQ.1) GO TO 995

C------MAIN ITERATION LOOP--PRINT STATUS
   DO 200 NC=1,NOCOL
      DO 250 NC=1,NOCOL
         C(NC)=C(NC)-COLIN(NC)*A(NR,NC)
      250 CONTINUE
      WRITE(6,193) NC
      IF(C(NC),GT.CMIN) GO TO 260
      IF(C(NC),GE.CMIN) GO TO 260
      C(NC)=C(NC)+COLIN(NC)*A(NR,NC)
      193 FORMAT('NO STARTING BASIC SOLN FOR ROW ',I5)
      195 CONTINUE
      IF(N03Sw.EQ.1) GO TO 995
      C------CHECK OPTIMALITY AND--OR FIND INCOMING COLUMN
      DO 205 NC=1,NOCOL
         IF(C(NC).GE.EPS) GO TO 210
      210 CONTINUE
      IF(C(NC).LT.EPS) GO TO 260
      NC=NC+1
      IF(NC.IE.MAXCOL) GO TO 210
      NOITER=1
      NOITER=NOITER+1
      WRITE(6,193) NC
      IF(C(NC),LE.EPS) GO TO 260
      193 FORMAT('NO STARTING BASIC SOLN FOR ROW ',I5)
      195 CONTINUE
      IF(N03Sw.EQ.1) GO TO 995
      C------MAIN ITERATION LOOP--PRINT STATUS
      DO 200 NC=1,NOCOL
         DO 250 NC=1,NOCOL
            C(NC)=C(NC)-COLIN(NC)*A(NR,NC)
         250 CONTINUE
         WRITE(6,193) NC
         IF(C(NC),GT.CMIN) GO TO 260
         IF(C(NC),GE.CMIN) GO TO 260
         C(NC)=C(NC)+COLIN(NC)*A(NR,NC)
      193 FORMAT('NO STARTING BASIC SOLN FOR ROW ',I5)
      195 CONTINUE
      IF(N03Sw.EQ.1) GO TO 995
      C------CHECK OPTIMALITY AND--OR FIND INCOMING COLUMN
      DO 205 NC=1,NOCOL
         IF(C(NC).GE.EPS) GO TO 210
      210 CONTINUE
      IF(C(NC).LT.EPS) GO TO 260
      NC=NC+1
      IF(NC.IE.MAXCOL) GO TO 210
      NOITER=1
      NOITER=NOITER+1
      WRITE(6,193) NC
      IF(C(NC),LE.EPS) GO TO 260
      193 FORMAT('NO STARTING BASIC SOLN FOR ROW ',I5)
      195 CONTINUE
      IF(N03Sw.EQ.1) GO TO 995
      C------MAIN ITERATION LOOP--PRINT STATUS
      DO 200 NC=1,NOCOL
         DO 250 NC=1,NOCOL
            C(NC)=C(NC)-COLIN(NC)*A(NR,NC)
         250 CONTINUE
         WRITE(6,193) NC
         IF(C(NC),GT.CMIN) GO TO 260
         IF(C(NC),GE.CMIN) GO TO 260
         C(NC)=C(NC)+COLIN(NC)*A(NR,NC)
      193 FORMAT('NO STARTING BASIC SOLN FOR ROW ',I5)
      195 CONTINUE
      IF(N03Sw.EQ.1) GO TO 995
      C------CHECK OPTIMALITY AND--OR FIND INCOMING COLUMN
      DO 205 NC=1,NOCOL
         IF(C(NC).GE.EPS) GO TO 210
      210 CONTINUE
      IF(C(NC).LT.EPS) GO TO 260
      NC=NC+1
      IF(NC.IE.MAXCOL) GO TO 210
      NOITER=1
      NOITER=NOITER+1
      WRITE(6,193) NC
      IF(C(NC),LE.EPS) GO TO 260
      193 FORMAT('NO STARTING BASIC SOLN FOR ROW ',I5)
      195 CONTINUE
      IF(N03Sw.EQ.1) GO TO 995
      C------MAIN ITERATION LOOP--PRINT STATUS
      DO 200 NC=1,NOCOL
         DO 250 NC=1,NOCOL
            C(NC)=C(NC)-COLIN(NC)*A(NR,NC)
         250 CONTINUE
         WRITE(6,193) NC
         IF(C(NC),GT.CMIN) GO TO 260
         IF(C(NC),GE.CMIN) GO TO 260
         C(NC)=C(NC)+COLIN(NC)*A(NR,NC)
      193 FORMAT('NO STARTING BASIC SOLN FOR ROW ',I5)
      195 CONTINUE
      IF(N03Sw.EQ.1) GO TO 995
      C------CHECK OPTIMALITY AND--OR FIND INCOMING COLUMN
      DO 205 NC=1,NOCOL
         IF(C(NC).GE.EPS) GO TO 210
      210 CONTINUE
      IF(C(NC).LT.EPS) GO TO 260
      NC=NC+1
      IF(NC.IE.MAXCOL) GO TO 210
      NOITER=1
      NOITER=NOITER+1
      WRITE(6,193) NC
      IF(C(NC),LE.EPS) GO TO 260
      193 FORMAT('NO STARTING BASIC SOLN FOR ROW ',I5)
      195 CONTINUE
      IF(N03Sw.EQ.1) GO TO 995
      C------MAIN ITERATION LOOP--PRINT STATUS
      DO 200 NC=1,NOCOL
         DO 250 NC=1,NOCOL
            C(NC)=C(NC)-COLIN(NC)*A(NR,NC)
         250 CONTINUE
         WRITE(6,193) NC
         IF(C(NC),GT.CMIN) GO TO 260
         IF(C(NC),GE.CMIN) GO TO 260
         C(NC)=C(NC)+COLIN(NC)*A(NR,NC)
      193 FORMAT('NO STARTING BASIC SOLN FOR ROW ',I5)
      195 CONTINUE
      IF(N03Sw.EQ.1) GO TO 995
      C------CHECK OPTIMALITY AND--OR FIND INCOMING COLUMN
      DO 205 NC=1,NOCOL
         IF(C(NC).GE.EPS) GO TO 210
      210 CONTINUE
      IF(C(NC).LT.EPS) GO TO 260
      NC=NC+1
      IF(NC.IE.MAXCOL) GO TO 210
      NOITER=1
      NOITER=NOITER+1
      WRITE(6,193) NC
      IF(C(NC),LE.EPS) GO TO 260
      193 FORMAT('NO STARTING BASIC SOLN FOR ROW ',I5)
57  CIN=NC(nc)
58  INCOL=NC
59  260 CONTINUE
60  IF(INCOL.EQ.0) GO TO 800
61  C-----PICK ROW TO PIVOT ON
62  INROW=0
63  RATM:=9999999,
64  DO 280 NR=1,NOROW
65  IF(A(NR,INCOL).LT.EPS) GO TO 260
66  RAT:=NR,NORMS)/A(NR,INCOL)
67  IF(RAT.LE.RATM) GO TO 280
68  RATM:=RAT
69  INROW:=NR
70  280 CONTINUE
71  IF(INROW.EQ.0) GO TO 300
72  C-----UNBOUNDED SOLUTION
73  WRITE(*,285) INCOL
74  285 FORMAT(3,0 SOLUTION UNBOUNDED--ADDNG COL'S IS)
75  GO TO 999
76  C-----PIVOT
77  300 INC=KAV(INROW)
78  KAV(INROW)=INCOL
79  DO 305 NR=1,NOROW
80  305 COLIN(NR)=A(NR,INCOL)
81  CSTIN=INC
82  COEF=KAV(INROW,INCOL)
83  DO 330 NC=1,NORHS
84  A(INROW,NC)=A(INROW,NC)/COEF
85  CHER=KAV(INROW,NC)
86  DO 310 NR=1,NOROW
87  IF(A(NR,EQ.INROW) GO TO 310
88  A(NR,NC)=A(NR,NC)-COLIN(NR)*CSTIN
89  310 CONTINUE
90  INC=INC-CSTIN-CORR
91  330 CONTINUE
92  310 CONTINUE
93  C-----END MAIN ITERATION LOOP
94  GO TO 200
95  800 DO 840 IL=1,NCOL
96  DO 820 IM=1,NOROW
97  810 VCTOR(Il,IL)=A(IL,NORHS)
98  820 CONTINUE
99  830 IF(NBASIC.EQ.0) GO TO 840
100  VCTOR(Il,IL)=0.
101  840 CONTINUE
102  DO 850 IN=1,NOROW
103  DO 850 IN=1
104  DUAL(IN)=999999,-C(ND)
105  850 CONTINUE
110  850 GO TO 999
111  C-----ERROR ROUTINES
112  910 WRITE(*,911)
113  911 FORMAT(3,0 TOO MANY ROWS OR COLUMNS)
GO TO 995
920 WRITE(6,921) NR
921 FORMAT(*0 NEGATIVE RHS IN ROW**I5)
GO TO 995
995 WRITE(6,996)
996 FORMAT(*0****RUN ABORTED*****)
999 RETURN
END
SUBROUTINE SPATH(INDX)
COST(25,26),DUAL(45),LABEL(25,150),
VECT(5),,V(45,150),VECT(50,150),J,I,K,L",",J,M,J,N,TOT,ISHORT(222),COSTM(25,26),"MIT,CTOTAL,CYNEG
DIMENSION SLABEL(25),ILABEL(25),IPERM(25),ILABEL(25),IPERM(25),IWAIT(27),DELTA(27,27)
ISTART=I
IF(0=1) CYNEG=0
II SAVE=0999
SCHECK=9999,
DO 14 II=1,M
SLABEL(II):=COSTM(ISTART,II)
ILABEL(II):=ISTART
IPERM(II)=0
IF(SLABEL(II),GE,SCHECK) GO TO 14
SCHECK=SLABEL(II)
II SAVE=II
14 CONTINUE
ILABEL(ISTART)=9999
IPERM(ISTART)=1
ICOUNT=1
IPERM(II SAVE)=1
ICOUNT=ICOUNT+1
DO 34 II=1,M
DO 34 IJ=1,M
DELTA(II,IJ)=0
34 CONTINUE
DO 54 II=1,M
IWAIT(II)=0
54 CONTINUE
II SAVE=II
DO 74 II=1,M
IF(IPERM(II),EQ,1) GO TO 74
C=CHECKS(II SAVE,II)+SLABEL(II SAVE)
IF(C,GE,SLABEL(II)) GO TO 74
SLABEL(II):=C
11. EL(II)=II SAVE
74 CONTINUE
DO 94 II=1,M
DO 84 IJ=1,M
IF(II,EQ,1.AND,IPERM(IJ),EQ,1.AND,IWAIT(IJ),EQ,0)
DELTA(II,IJ):=SLABEL(II)+COSTM(II,IJ)-SLABEL(IJ)
TEST=1M=0.0005
IF(DELTA(II,IJ),GE,TEST) GO TO 84
1I M=DELTA(II,IJ)
MINT=II
MINL=IJ
84 CONTINUE
DO 94 II=1,M
IWAIT(IJ)=I
94 CONTINUE
IF(DM INT,GE,0.) GO TO 9A
1=IWAIT(MINL)=1
SLABEL(MINL)=SLABEL(MINT)+COSTM(MINT,MINT,MINT,MINT)
57       ILABEL(MINL)=MINT
58       CYAEG=SLABEL(MINL)+COST(MINL,IGADD)+SLABEL(IGADD)
59       IF(CYAEG.LT.0.) GO TO 96
60       II SAVE=MINL
61       DO 95 II=I,II
62       DELTIA(II,MINL)=0.
63       95 CONTINUE
64       GO TO 64
65       SLABEL(IGADD)=COST(MINL,IGADD)+SLABEL(MINL)
66       ILABEL(IGADD)=MINL.
67       ILOOP=I3ADD
68       GO TO 105
69       96 II=II+2
70       IF(I COUNT .LT.0.1) GO TO 104
71       SCHECK=9999.
72       IISAVE=9999
73       DO 99 II=I0ADD
74       IF(IP .LT.(II).E0.1) GO TO 99
75       IF(SLABEL(II).GE.SCHECK) GO TO 99
76       SCHECK=SLABEL(II)
77       IISAVE=II
78       99 CONTINUE
79       GO TO 24
80       104 ILOOP=IEND
81       105 II=II+2
82       DO 114 II=IJ,II,-1
83       ISHORT(II)=ILOOP
84       IILAST=II
85       ILOOP=ILABEL(ILOOP)
86       IF(CYAEG.LT.0..AND.ILOOP.EQ.IGADD) GO TO 115
87       IF(CYAEG.GE.0..AND.ILOOP.EQ.ISTART) GO TO 115
88       114 CONTINUE
89       115 ISHORT(II)=INDX
90       II=IILAST
91       IF(CYAEG.LT.0.) IX=IILAST+1
92       IF(CYAEG.GE.0.) ISHORT(IX)=I3ADD
93       DO 120 II=2,II
94       IILAST=II
95       ISHORT(II)=ISHORT(IX)
96       IF(IK.EQ.IJ) GO TO 122
97       IFFK+1
98       120 CONTINUE
99       122 IF(CYAEG,GE,0.) ISHORT(IILAST)=INDX
100      II=ISHORT(2)
101      CTO TAL=COST(INDX,II)
102      DO 124 II=3,IILAST
103      I=II-1
104      I=ISHORT(II)
105      I=ISHORT(IK)
106      CTO TAL=CTOTAL+COST(IK,II)
107      124 CONTINUE
108      125 IF(CYAEG,GE,0.) GO TO 138
109      135 CTO TAL=CTOTAL
110      DO 138 II=3,II
111      I=II+1
112      IV=ISHORT(II)
113      I=ISHORT(IK)
114  CTOTAL=CTOTAL+COST(IW,IV)
115  IF (IW=0, ILOOP) GO TO 138
116  CONTINUE
136  CONTINUE
118  II=IILAST+1
119  IK=J+2
120  DO 144 IU=II,IK
121  ISHORT(IJ)=0
122  CONTINUE
144  CONTINUE
123  IF (ISHORT(2),EQ,INDX) CTOTAL=0.
124  IF (CYNES.LT.0.) RETURN
125  IF (CYL.SZ.0..AND.SLABLE(INEND).LE.0.) RETURN
126  ISHORT(2)=INDX
127  ISHORT(3)=INDX
128  CTOTAL=0.
129  RETURN
130  END
Data Generation User's Instructions

The test data generation was accomplished by the following two subroutines being incorporated into the main program. The 70 and 80 DO-loops of MAIN were replaced by a CALL GENERA card.

The only data cards now required are:

(i) The number of depots, the number of jobs as integers separated by a comma.
(ii) The random number seed, which must be a four digit odd integer.
(iii) A code card with 0 in the first column to inactivate subroutine MODIFY.
DATA GENERATION SUBROUTINES

FORTRAN SOURCE CODE LISTING
SUBROUTINE GENERA
C
COMMON (C(150),SAVE(150),COST(25,26),DUAL(45),LABEL(25*150),
INVEST(5),RMP(45,150),VECTOR(50,150),I,J,K,L,*,J,M,JN,NTOT,ISHORT(1
222),COSTM(25,26),AMIT,CTOTAL,CYNEG)
INTEGER LABEL,D
READ(*,11)ISEED
11 FORMAT(1)
IC=J/0
NTC=1
DO 41 II=1,0
INTAL=0
DO 31 IJ=1,IC
DO 21 IK=1.D
JIST7(II—IK)*50
IF(DIST.LT.0.) DIST=—DIST
11 INT=50.*DRAND(ISEED)+DIST
IF(IN1T.LT.0.) INT=—INT
COST(IK,NTC)=INT
IF(DRAM(ISEED)+LT..1) INT=INT+5
COST(NTC,IK)=INT
CONTINUE
INT=10.*DRAND(ISEED)
IF(IN1T.LT.0.) INT=INT
COST(IN1T,L)=INT
NTC=INTAL+INT
NTC=NTC+1
CONTINUE
INTAL=INTAL+2*INT
AL=INTAL
COST(I,I,L)=INTAL
CONTINUE
DO 61 II=1,K
DO 51 IJ=IPT
IF (II.GT.IJ) GO TO 51
IK=II—IJ
IK=—IK
IK=0
JIST4,5C
IT:=30.*OND(ISEED)+DIST
IF(IN1T.LT.0.) INT=—INT
COST(I,II,L)=INT
CC 5 T(IJ,II)=9999.
51 CONTINUE
61 CONTINUE
WRITE(6,71) D,J
71 FORMAT(11,9(/)14X,A*)11,1,1, DEPOT VEHICLE ASSIGNMENT PROBLEM WI
14*,112, JOB LOCATIONS*2(/)14Y, INITIAL COST MATRiX*1(/)
WRITE(6,72)((II),II=1,K)
72 FORMAT(14X,1115*1(/))
51 DO 74 IJ=1,K
52 WRITE(6,71) II,(COST(I1IJ),IJ=1,K)
53 WRITE(6,71) II,(AVAILABILITIES AND DEMANDS 
54 74 CONTINUE
55 WRITE(6,75)
56 75 FORMAT(0*,13X,'AVAILABILITIES AND DEMANDS')
WRITE(*,75)(COST(I,I),I=1,K)
76 FORMAT(16X,11F5.0)
RETURN
END

FUNCTION DRAND(ISEED)
    ISEED=ISEED*131072
    IF(ISEED.LE.0) ISEED=ISEED+34359738366741
    DRAND=ISEED*2910383E-10
RETURN
END
APPENDIX B

Dual Value Calculations

This appendix will examine the means of calculating the restricted master dual variable vector, \( \pi \). Consider a basis, \( B \), and let \( A = [B|N] \). Partitioning \( C \) and \( X \) vectors, we get the problem in the following form:

\[
\begin{align*}
\text{Min } & \quad Z_o = C_B X_B + C_N X_N \\
\text{s.t. } & \quad B X_B + N X_N = b
\end{align*}
\]

Now if we premultiply the constraint by \( B^{-1} \) we get \( X_B = B^{-1} b - B^{-1} N X_N \). Eliminating \( X_B \), the objective function becomes

\[
C_B B^{-1} b - C_B B^{-1} N X_N + C_N X_N
\]

We can rewrite this expression as

\[
C_B B^{-1} b + (C_N - C_B B^{-1} N) X_N,
\]

where

\[
C_B B^{-1} b \text{ is the value of } Z_o \text{ and } (C_N - C_B B^{-1} N) \text{ gives the coefficients of the non-basic variables (given in row zero). Since } C - C_B B^{-1} A = C - \pi A \text{ where } \pi \text{ is the dual vector,}
\]

\[
C - C_B B^{-1} A = C - \pi A
\]
Now consider the identity matrix obtained from the columns of the basic variables in the initial tableau. Let $C_I$ be its coefficients in the objective function and let $C'_I$ be its updated coefficients. Then

$$C'_I = C_I - \pi I \quad \text{or} \quad \pi = C_I - C'_I.$$
BIBLIOGRAPHY


