

ON THE VEHICLE SCHEDULING MODEL
AND ASSOCIATED PROBLEMS

A THESIS

Presented to

The Faculty of the Division of Graduate
Studies and Research

By

Ivan Jose Perez C.

In Partial Fulfillment
of the Requirements for the Degree
Master of Science
in Industrial and Systems Engineering

Georgia Institute of Technology

December 1975

ON THE VEHICLE SCHEDULING MODEL
AND ASSOCIATED PROBLEMS

Approved

[Handwritten signature]

Dr. Robert G. Parker, Chairman

[Handwritten signature]

Dr. Douglas C. Montgomery ✓

[Handwritten signature]

Dr. Paul Jones

Date Approved by Chairman: 11/25/75

ACKNOWLEDGMENTS

I would like to express my appreciation to Dr. Robert G. Parker who served as my thesis advisor. Dr. Parker contributed a great deal of needed insight to this research effort, and his personal efforts helped to bring it to this written conclusion. I am also grateful to my reading committee members, Dr. P. Jones and Dr. D. C. Montgomery for their helpful comments.

The help provided by Mr. Carlos Isaza in the computer program is greatly appreciated.

This thesis is dedicated to my mother, Carmen. The encouragement through her letters was invaluable to me during my graduate studies.

TABLE OF CONTENTS

	Page
ACKNOWLEDGMENTS	ii
LIST OF TABLES	iv
LIST OF ILLUSTRATIONS	vi
Chapter	
I. INTRODUCTION	1
Problem Definition	
Literature Review	
Organization of Study	
II. A HEURISTIC ALGORITHM FOR THE TRAVELING SALESMAN PROBLEM	14
Introduction to Holmes' Algorithm	
A Saving Procedure for the Traveling Salesman Problem	
Computational Algorithm	
Sample Problems	
Computational Experience	
III. DELIVERY PROBLEM	36
The Basis for an Approach	
Computational Algorithm	
Sample Problems	
Computational Experience	
IV. CONCLUSION AND EXTENSTIONS	75
APPENDIX	70
REFERENCES	98

LIST OF TABLES

Table	Page
2.1. Original Cost Matrix of Small Euclidean Problem	22
2.2. Cost Matrix Using City (2) as a Depot	22
2.3. Saving Matrix of Euclidean Problem	24
2.4. Current Final Solution Matrix	24
2.5. Suppression Number Two (2) of $K = 152$	25
2.6. Cost Matrix for Non-Symmetric Problem	26
2.7. Saving Matrix for Non-Symmetric Problem	26
2.8. Current Final Solution Matrix	27
2.9. Final Solution Matrix for Suppression # 1 of $K = 21$	28
2.10. Comparative Result of Published Problems	33
3.1. Data for Sample Problem; Cost Matrix and Demand Requirements	37
3.2. Initial Saving Matrix	37
3.3. The First Step Solution Matrix	38
3.4. Intermediate Solution Matrix	39
3.5. Final Solution Matrix	40
3.6. The First Step Solution Matrix Using Link (6-3) as a Starting Point	41
3.7. Intermediate Solution Matrix Using Link (6-3) as a Starting Point	41
3.8. Final Solution Matrix Using Link (6-3) as a Starting Point	42
3.9. Saving Matrix for a Symmetric Example	50

Table	Page
3.10. Data for a Sempel Problem Case I (Cost Matrix; Demands and Saving Matrix)	51
3.11. Initial Reduced Matrix	53
3.12. Intermediate Reduced Matrix	53
3.13. Final Solution Routes (Clark and Wright's Solution) . . .	54
3.14. Initial Reduced Matrix for the New Starting Link (7-2) . .	55
3.15. Intermediate Solution Matrix	55
3.16. Final Solution Matrix for Link (7-7)	56
3.17. Data for a Sample Problem Case II	58
3.18. Final Solution Matrix for Truck Type I	59
3.19. Initial Saving Matrix for Truck Type 2	60
3.20. Final Solution Matrix Taking Link (2-4) as Starting Joined Pair	61
3.21. Summary of Small Problems Existing in the Literature . . .	63
3.22. Summary of Multi-Salesman Problems	65
3.23. Percentages of Importance over Clark and Wright Procedure	66
3.24. Summary of Symmetric Truck-Delivery Problems	69
3.25. Summary of Random Generated Problems	72

LIST OF ILLUSTRATIONS

Figure	Page
2.1. Demand Point Before and After Being Joined on a Route . . .	15
2.2. Summary of the Complete Solution for City (1) as a Depot	29
2.3. Final Solution for Every City Taken as a Depot	30
2.4. Map of the 42-City Problem	32
3.1. Summary of Different Solutions	57
3.2. Solution of 2 Multi-Salesman Problem Based Upon a Single-Salesman Tour	64
3.3. Computation Time of Multi-Salesman Problems	67
3.4. Computation Time of Symmetric Truck Delivery Problems . . .	70
3.5. Computation Time of Non-Symmetric Randomly Generated Problems	73

CHAPTER I

INTRODUCTION

Problem Definition

A popular class of problems in Operation Research consists of those which possess combinatorial characteristics. Characterized mainly by the achievement of particular arrangements or ordering of elements into sets, combinatorial problems have long been among the most troublesome from a computational view. As such, many problems of practical size remain void of solution even in the present day of powerful and accessible computing machinery.

This thesis is concerned with a well known combinatorial problem as well as certain of its related models. In particular, a problem of vehicle delivery is studied such that an approach to its solution is formulated. In addition, special cases of the delivery problem are addressed among which is the celebrated traveling salesman problem. Following, fundamental aspects and definitions of the primary problem are addressed, as are relevant assumptions which lead to the related problems. A pertinent literature review is presented and, finally, the general organization of the thesis is presented.

The general problem has been considered frequently in the literature and is known as the vehicle scheduling problem, the transportation routing problem or even the truck delivery problem. Simply stated, the problem is one of routing a fleet of vehicles of known

capacities from a central "depot" or terminal to a set of demand points and back to the terminal. The merchandise is homogeneous, the shortest route between every two points is given and the demand for each point is assumed known. The objective is to allocate loads to trucks in such a way that all merchandise is delivered and the total distance traveled is a minimum. No restriction is placed upon the time or interval in which the demand must be serviced.

Manifestations of the delivery problem are common. One example would be the daily delivery of bread from a bakery to retail stores; another would be the delivery of fuel oil to gas stations. In addition closely related problems, such as the determination of the smallest number of vehicles required to meet a given demand, efficient use of an airplane fleet to meet the daily schedules, garbage collection, taxi fleet utilization and police patrol car deployment could all benefit from analyses dealing with the so-called "delivery problem".

It is also of interest to note that the delivery problem is a generalization of two other well-known problems. If one assumes that the demand of all customer is equal to one and that vehicle capacity is k units ($k < n$), then the multi-salesman problem results. Continuing, suppose that vehicle capacity is at least as large as to total demand. In such a case, the ordinary traveling salesman problem results.

Literature Review

In recent years, the transportation routing problem as well as its special cases mentioned above, has attracted the attention of many

researchers. As a consequence, numerous appearances in the literature have resulted. A survey of the meaningful work follows, organized according to the approach taken: (1) Integer Linear Programming, (2) Dynamic Programming, (3) Branch and Bound, and (4) Heuristic.

Integer Linear Programming

Integer linear programming is one of the exact solution procedures that has been frequently used to model combinatorial problems. Perhaps the first appearance of such a procedure relative to the delivery problem was by Balinski and Quant [3]. In essence, a version of Gomory's cutting plane method [17] was employed. Alternately, Garving, Crandall, John and Spellman [14] gave an entirely different linear programming formulation for the problem. Consider the following:

Y_{iju} = number of unit shipped from i to j defined for k ,

D_k = demand at k , x_{ij} = number of truck runs from i to j

g = truck capacity,

C = distance from i to j (cost)

$$(1) \sum_i Y_{ijk} = \sum_{\mu} Y_{j\mu k} \quad \forall j, k \text{ and } j \neq k$$

$$(2) \sum_i Y_{ikk} = D_k, \quad \forall k$$

$$(3) \sum_k \sum_j Y_{ojk} = \sum_k D_k$$

(1) Y_{ijk} = quantity that arrives at j from all points destined for k

Y_{iuk} = sum of what leaves j for all points destined for k

(must be equal since no accumulation at j for k)

(2) What arrives at k destined for k = demand at k

(3) What leaves warehouse - total demand

If a truck enters j it must leave j and only one truck will

serve j

$$\sum_i X_{ij} = \sum_u X_{ju} = 1 \quad X_{ij} = 0 \text{ or } 1$$

Capacity constraint on trucks

$$\sum_k Y_{ijk} \leq gX_{i,j} \quad \forall i, j \quad i \neq j$$

The objective function is Min: $\sum_i \sum_j C_{ij} X_{ij}$

This formulation requires on the order of n^3 variables and n^2 constraints, for $n = 10$ this gives 1000 variables and 100 constraints which is clearly impractical. Relative to the special case of the traveling salesman problem, a well known integer program of Miller, Tucker and Zemlin [26] can be given as follows:

$$\text{Min } Z = \sum_{i=1}^n \sum_{j=1}^n d_{ij} X_{ij}$$

ST.

$$\sum_{i=1}^n X_{ij} = 1 \quad j = 1, 2, \dots, n$$

$$\sum_{j=1}^n X_{ij} = 1 \quad i = 1, 2, \dots, n$$

$$Y_i - Y_j + nX_{ij} \leq n - 1 \quad \forall i \neq j \text{ and } i, j \notin I_0 (I_0 = 1)$$

$$X_{ij} = 0, 1 \quad \forall i, j$$

Where:

$Y_i, i = 1, \dots, n$ are arbitrary numbers

$X_{ij} =$
 1 if the salesman travels directly from city i to city j
 0 otherwise.

$d_{ij} =$ direct distance from city i to city j .

For large values of n (the number of cities) this formulation becomes unwieldy. Indeed, few traveling salesman problems are solved by integer programming.

Dynamic Programming

Using dynamic programming, an optimal solution to the transportation routing problem can be developed by reducing the problem to that of finding the shortest path through a network. The solution algorithm is a modification of the one stated below and is discussed in reference [34].

The use of dynamic programming as an approach to solving the traveling salesman problem has been suggested by Bellman [4]. In his formulation, the home base or start of the tour can be defined as any city, say city 1. Suppose at some stage in an optimal tour starting at city 1, city i has been reached and there remain k cities j_1, j_2, \dots, j_k to be visited before returning to 1. Then, it is clear that if the tour is to be optimal, the path from i through j_1, j_2, \dots, j_k in some

order and then to 1 must be of minimal length. Consider the following:

$C(i; j_1, j_2, \dots, j_k)$ = the length of a path of minimum length from i to 1 which passes exactly once through each of the remaining k unvisited cities j_1, j_2, \dots, j_k .

Thus, $C(1; 2, 3, \dots, N)$ and the path that has this as its length, constitutes a solution to the problem. Now,

$$C(i; j_1, \dots, j_k) \leq d_{ij_m} + C(j_m; j_1, j_2, \dots, j_{m-1}, j_{m+1}, \dots, j_k) \quad m = 1, 2, \dots, k$$

By the definition of $C(i; j_1, j_2, \dots, j_k)$:

$$C(i; j_1, \dots, j_k) = \min \{ d_{ij_m} + C(j_m; j_1, \dots, j_{m-1}, j_{m+1}, \dots, j_k) \}$$

for $k = 1$

$$C(i; j_1) = d_{ij_1} + d_{j_1}$$

The last two equations are applied recursively to solve the problem. As is often the case, the dynamic programming approach requires a great deal of storage. Consequently, its use on large problems is limited if not impossible altogether.

Branch and Bound

Branch and bound is essentially a tree-search in which each branching of the tree from a node represents a partitioning of the set

of all possible solutions into those that contain a particular link or city-pair and those that do not. Associated with each node thus formed is a number which is a lower bound on the cost of any solution involving the decision at that node. The branching process continues until a complete solution is found whose cost is less than or equal to the lower bounds of all other "unbranched from" nodes, regardless of their level in the tree. One of the earliest studies dealing with branch and bound was carried out by Little [25] such that a landmark algorithm for dealing with the traveling salesman problem resulted.

An extension of Little's approach was developed by Hayes [18] in order to treat the delivery problem. It was reported however, that the method appears to be unsuitable for solving problems of even moderate size (greater than fifteen customers).

Another exact technique to date is that developed by Svestka and Huckfeldt [31]. Their procedure is designed to solve the closely related m-traveling salesman problem. The procedure first solves a related assignment problem using a modified transportation algorithm of Ford and Fulkerson [13]. If the solution to the modified assignment does not satisfy the conditions of the m-salesman problem, a restriction is imposed upon the cost matrix, say D . These restrictions prohibit the formation of the previous solution but do so without prohibiting any feasible solution. In this method the assignment problem is solved successively until all conditions are satisfied. Forming a branch of a solution tree with the sequence of assignment problem a branch and bound technique is used to guarantee optimality. In [31], the foundation for the branch and bound scheme is from a development

by Bellmore and Malone [5]. However, the formulation, the initial tour generation, and the generalization of the algorithm to the m -salesman case are original. Due to the generalization, the branching is not quite the same as the Bellmore and Mallone scheme for any number of salesmen. As with the most branch and bound schemes, computation time increase exponentially with the number of cities. However, the algorithm matches the computational times of Bellmore and Malone in the one salesman case. Further it is reported that the inclusion of more than one salesman does not make the problem more difficult; and in fact reduces computational effort.

A slight variant of the spanning tree problem was used by Held and Karp [19] to solve the related traveling salesman problem. It is shown that a minimum-weight 1-tree can be found by constructing a minimum spanning tree on the vortex set $\{2, \dots, n\}$ and then adjoining two edges of lowest weight at vortex 1. This is important because every tour is a 1-tree, and 1-tree is a tour if and only if each of its vortices has degree 2. Therefore, if a minimum-weight 1-tree is a tour, it is the solution to the traveling salesman problem. Held and Karp introduce a "gap" function $f(\pi)$ equal to the amount by which the cost of a minimum tour with respect to the weights $c_{ij} + \pi_i + \pi_j$ exceeds the cost of a minimum 1-tree with respect to the same weights. Thus, $f(\pi)$ is a non negative function that measures the size of this gap, and assumes the value of 0 at its minimum when it is possible to obtain a minimum 1-tree that is a tour. The problem is left to minimize $f(\pi)$. One of the approaches used to minimize that function was branch

and bound, using as a fundamental element the concept of an out-of-kilter vortex. No computer computation with the branch and bound method was reported.

Heuristic

The inability of the optimal-seeking procedures to solve reasonably sized traveling salesman problems and truck delivery problems in a feasible amount of time has necessitated the development of alternative methods which take the form of heuristics. The earliest such paper on the delivery problem was published by Dantzig and Ramser [10] in 1959. In the procedure each customer is first assigned a route for itself. Aggregation into pairs is then performed making use of the distance matrix to detect likely links, based on the proximity of any two particular customers. If a link is formed and subsequently seems disadvantageous, it may be broken and a new one found. Once this aggregation has been achieved, a new distance matrix is formed using the aggregation in a manner analogous to customers in the previous step. This process continues until all aggregations are completed. At the first stage of aggregation no pairing is allowed which results in a combined demand greater than $(C/2)^{N-1}$ where N is the total number of stages which will be required. Observing this constraint in the first stage allows all further aggregations to be made without concern for the capacity constraint.

The method of selection for the joining of points does not guarantee optimality, but the technique has served as basis for many heuristic solution procedures existing currently.

In an effort to construct improved solutions to the delivery

problem, Clarke and Wright [6] developed a heuristic algorithm which in some respects is quite similar to Dantzig and Ramser's. Clarke and Wright start with each customer assigned a route of its own. Routes are then combined by maximizing the "savings" achieved by joining two of these routes (always subject to the capacity constraint). The final solution is then built up by successive recombination of smaller routes and single customers. This procedure will be discussed in Chapter II and III in greater detail, since it is used as a basis for this work.

Tillman and Cochran [32] extended the work of Clark and Wright such that a "look-ahead" procedure was developed. They chose the arc to be added as that link which allows the second best to be chosen so that the sum of the two is largest. This could be extended to a look-ahead depth of three or more.

Gaskell [15] also proposed a modification of Clarke and Wright method. He suggested changing the calculation of savings to give more consideration to unusual C_{ij} 's. His savings are calculated as one of the below:

$$\lambda_{ij} = S_{ij} (\bar{C} + |C_{oi} - C_{oj}| - C_{ij})$$

$$\pi_{ij} = S_{ij} - C_{ij}$$

Where \bar{C} = average value of all C_{oi} . The remainder of the procedure is the same as Clarke and Wright.

Christofides and Eilon [9] took a somewhat different approach. Their "r-optimal" procedure was based upon work done by Lin [23]

concerning to the traveling salesman problem. This approach starts with a feasible solution and tests perturbation to obtain r optimality. If $r = 2$, each pair of arcs are examined to see if they can be replaced by another pair of arcs such that the routes are still feasible and more economical. This procedure can be used with any of the other heuristic procedures since it does require an initial feasible solution. They report, in general, an improvement in the solution quality, but at expense of greater computation time.

A recent paper by Newton and Thomas [28] has employed yet another solution approach to the delivery problem. The method consists of finding a solution to the traveling salesman problem having the same cost matrix and then partitioning the one traveling salesman route into individual truck routes which conform the capacity constraints. In Chapter III the weakness of this approach is demonstrated.

The most recent extension of the Clarke and Wright procedure has been done by Holmes [21]. He points out that the selection of pair for joining entirely on the basis of savings may cause vehicles to be loaded at substantially low capacities, or it may prohibit pair joining at later stages which would otherwise result in greater total savings. The suggested improvement is to "suppress" the joining of certain demand points that alone yield high savings, but which affect adversely subsequent joinings. This method has proved to be computationally desirable in that solutions of good quality are attained for problems of relatively large size. This approach is shown in Chapter II to be suitable for a heuristic solution to the traveling salesman problem.

Relative to the traveling salesman problem, a number of heuristic solutions have been proposed. Karg and Thompson [22] proposed an approach utilizing a shortest path distance matrix. This matrix is then used to generate an optimal or near optimal route. Ashour, Vega, and Parker [1] presented an algorithm which in essence is a branch and bound approach without backtracking. Several ramifications in the algorithm, such as the application of two depth look-ahead rule, a modified regret function, and a saving function were attempted. The algorithm provides solution with good quality in a feasible amount of computer time.

Most recently, Lin and Kernigam [24] proposed a heuristic method that has proved to be one of the most successful to date. However, the method is good only for symmetric problems. It uses the same interchange philosophy as do many of the heuristics and employs a rationale very similar to the r-optimal concept. Basically, the procedure starts with a pseudo-random solution and continues the solution until no further improvement is available. It does this iteratively, however, so that it is similar to r-optimal but r is not fixed. The key to the success of the procedure is the following theorem.

"If a sequence of numbers has a positive sum, there is a cyclic permutation of these numbers such that every partial sum is non negative". This allows the procedure to consider only sequences of interchanges whose partial sum of the gains is always non negative. Computational experience is reported as very good. On small problems (up to 42 cities) the probability of obtaining an optimal solution is

close to one.

Organization of Study

This research deals with two extensions of the Clarke and Wright procedure. The first is developed in order to treat the classic, Euclidean traveling salesman problem and the second to treat the delivery problem. Chapter II is concerned with the traveling salesman problem such that an algorithm is discussed and presented. Suitable experience with the algorithm is given. Chapter III deals with the delivery problem. A detailed description of the method developed for its solution is given, including a computational statement of the algorithm. Again computational experience is provided. Finally, Chapter IV presents a summary of the applicability and competitiveness of the two approaches, relative to solution quality and computational effort. Areas for improvement and extensions are discussed.

CHAPTER II

A HEURISTIC ALGORITHM FOR THE TRAVELING SALESMAN PROBLEM

Introduction to Holmes' Algorithm

An algorithm is presented in this chapter which is based, fundamentally, upon previous work relative to the delivery problem [21]. In essence, an approach is pursued which is based upon the notion of savings [6] coupled with a specific solution disturbance scheme which has been shown previously to be effective in achieving solutions of good quality without great expenditure in computational effort. Consequently, the basic concepts of the previous algorithm are discussed after which its formalization is specified, demonstrated and tested relative to use for the special case of the traveling salesman problem.

A recent piece of research by Holmes [21], itself based on the previous work of Clarke and Wright [6], was devoted to the development of a functional algorithm for the vehicle scheduling problem. While the current chapter is concerned with the special case of the traveling salesman problem, the terminology of a truck delivery problem will be maintained while presenting certain basic concepts. Consider a feasible allocation of trucks to demand points shown in Figure 2.1(a). The demand points $P(1)$ and $P(2)$ are initially linked only to the terminal point $P(0)$. Two trucks, each traveling from the terminal point to a demand point and back to the terminal point, are allocated to haul the loads required by the demand points.

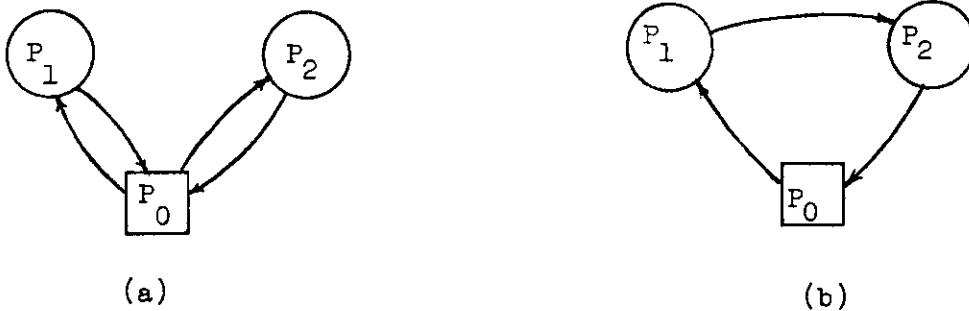


Figure 2.1. Demand Point Before and After Being Joined on a Route

The total distance traveled in Figure 2.1(a) is:

$$D = d_{01} + d_{10} + d_{02} + d_{20}$$

However, linking the two demand points P(1) and P(2) on a route and removing links from point P(1) to the terminal and from the terminal to point P(2), results in a single route shown in Figure 2.1(b). The total distance now becomes:

$$D' = d_{01} + d_{12} + d_{20}$$

The two solutions depicted differ in cost by an amount, say ΔC such that

$$\Delta C = D - D' = d_{01} + d_{10} + d_{02} + d_{20} - d_{12} - d_{20}$$

$$\Delta C = d_{10} + d_{02} + d_{12}$$

If $\Delta C > 0$, then there is realized a reduction in distance by combining the two demand points onto a single route. Note further, that in routes of both figures, directionality is maintained since in the

current work, symmetric as well as non-symmetric problems are considered.

The notion of savings is simple. Joining two points on a route will reduce the total cost required to service the points by an amount equal to the savings associated with the two points. Thus, the total cost after joining two points is equal to the total cost before joining, less the saving of the joined pair. That is:

$$\text{Cost (after)} = \text{Cost (before joining)} - \text{Total Savings}$$

This relation is true for each pair of points joined. It must then be true that the final solution is equal to the initial solution cost less the sum of all savings, which is obtained by joining points in the solution. That is

$$\text{Cost (final)} = \text{Cost (initial)} - \left[\sum_{i=2}^n \sum_{j=2}^n S_{i,j} X_{i,j} \right]$$

The same relation between initial and final cost holds for any solution. Thus, for the optimal solution it must be true that the total savings is maximized. A procedure suggests itself then such that all possible savings are calculated and tabulated in a matrix, after which pairs of the maximum savings feasible with respect to truck capacities are successively joined. This process continues until all possible joinings are made.

The above scheme is basically the procedure of Clarke and Wright. Holmes, among others however, has identified one of the major weaknesses of this method. It can be observed that in many optimal

solutions, a pair of points selected for joining by the Clarke and Wright procedure do not appear at all. It has been pointed out that selection of a pair for joining entirely on the basis of savings might prohibit joinings at later stages which ultimately result in greater savings.

To help overcome such a drawback Holmes suggested the suppression of certain links that alone yield high savings but which in the long run adversely affect subsequent joinings. The suppression notion can be summarized as follows: The problem is first solved using the basic solution procedure (Clarke and Wright), keeping a list of the order in which each joining was made. The first joined pair in the current solution is then suppressed and the modified problem solved again by the original procedure. If an improvement is obtained, the new solution is saved and the procedure continues, suppressing the first link of the new solution. If no improvement is obtained, the current suppression is neglected and the next joined pair is suppressed. The process continues until a predetermined number of successive suppressions yields no improvement or when all links in a current solution have been suppressed with no improvement. Holmes reported that in large problems, a maximum of five suppressions was found to be effective.

A Savings Procedure for the Traveling Salesman Problem

The solution to the traveling salesman and the transportation routing problems are very similar as both require a tour (or tours) passing through every city (demand point). Consequently, many solution procedures for the transportation routing problem are based on the

traveling salesman solution procedures. As suggested earlier, the traveling salesman problem can also be treated as a special case of the delivery problem. Such an approach is taken in this chapter.

Attempts to apply the savings method of Clarke and Wright to the traveling salesman problem, have been reported in the literature as unsuccessful [1]. Reasons suggested for such a conclusion are: (1) savings are fixed in value throughout the problem solving procedure (this means that there is no savings updated to remaining links after a decision is made), (2) the existence of the depot as a fixed location for reference is too restrictive to evaluate savings and, (3) theoretically, there exists a different solution for every city used as a central "depot". Assuming that one randomly selects a city as a depot and applies the savings approach, experience in this research tends to substantiate the above claims. A potential alleviation to such drawbacks is formulated.

For problems of small size, say 10 points or less, a selection of a specific point (or points) as depot appears to be of little consequence. In fact, the application of the saving scheme with solution perturbation (suppression) yielded optimal solutions with extreme frequency in such cases. However, as problem size increases a considerate choice of point(s) for the assumed depot becomes more important. Such criticality is evident in the necessity to achieve a "good" solution with reasonable effort where the latter cannot be affected, for example by simple specification of every point as a depot, application of the algorithm n times (n points) and selection

the best solution.

Suppose, under the assumption of Euclidean distances, that the points to be selected for the salesman itinerary are plotted on two coordinate axes. Further, let the centroid or center of mass be given by $(\bar{x}; \bar{y})$ such that

$$\bar{x} = \sum_{i=1}^n \frac{x_i}{n} ; \bar{y} = \sum_{i=1}^n \frac{y_i}{n}$$

Selection of the point nearest the centroid (or sets of points within some region near it) and specification of it as the depot which services the other $n-1$ points, might yield good results. After solution of four well-known test problems with $n = 25, 33, 42,$ and 57 , such a notion has proven relatively worthy. Specific results are given subsequently.

Computational Algorithm and Sample Problems

The original computational solution procedure given by Holmes will be repeated here with the modifications introduced to treat the traveling salesman problem. It should be mentioned that the algorithm is specified for symmetric problems, but it can be used for others - neglecting the notion of using the centroid to determine "depots".

Step 1 - Initialization

1.1 Construct an initial cost matrix C , such that $C = [C_{i,j}]$ for all points i, j .

1.2 Initialize a suppression counter L at 1, and the maximum suppression number, L'

1.3 Compute the centroid such that

$$\bar{x} = \sum_{i=1}^n x_i/n ; \quad \bar{y} = \sum_{i=1}^n y_i/n$$

and specify a set of cities S, nearest $(\bar{x}; \bar{y})$ where $|S| \geq 1$.

1.4 Take the first city in S and initialize a counter s at 1.

1.5 Change the cost matrix such that

$$C_{i_1} \leftarrow C_{ij} * \forall i$$

$$C_{1j} \leftarrow C_{i*j} \quad \forall j$$

"*" will represent a city in the set "S". Proceed to step 2.

Step 2 - Construct the saving matrix and the initial solution.

2.1 Compute $S_{i,j}$ such that

$$S_{i,j} = C_{i1} + C_{1j} - C_{ij} \quad \forall i, j = 2, 3, \dots, n \text{ and } i \neq j$$

Let $S_{i,j} = 0$ for all $i = j$

and set $S_{i,1} = S_{1,j} = -1 \quad \forall i, j = 2, 3, \dots, n$

2.2 Compute the cost of the initial solution such that

$$K = \sum_{i=2}^n C_{i,1} + \sum_{j=2}^n C_{1,j}$$

Step 3 - Determine a candidate pair

3.1 Find the ordered pair (\hat{i}, \hat{j}) with the greatest feasible savings such that

$$S_{i,j} = \frac{\text{Max}}{(i,j)} [s_{ij}]$$

Where $(\overline{i, j})$ is defined over all ordered pair such that $S_{i,1} = S_{1,j} \neq 0$

3.2 If $S_{\hat{i}; \hat{j}} = 0$ go to step 5

Step 4 - Join the points i and j on a route

4.1 If neither of the points is on a route construct a new route z and proceed to step 4.5.

4.2 If one of the points is on a route, say z, join the unassigned point to z proceed to step 4.5.

4.3 If both points are on a route subtour is formed. Set $S_{\hat{i};\hat{j}} = 0$; return to step 3.

4.4 If both points are currently assigned to routes u and v join both routes into one route z.

4.5 Compute the new solution cost such that $K \leftarrow K - S_{\hat{i};\hat{j}}$. Make the necessary up-to-date to the saving matrix such that

$$S_{\hat{i};\hat{j}} = -1; S_{\hat{j};\hat{i}} = 0; S_{\hat{i};\hat{l}} = S_{\hat{j};\hat{j}} = 0$$

Return to step 3.

Step 5 - Save the best solution

5.1 If this is the first solution, maintain the cost K' , such that $K' = K$. Keep all routes and the order in which points were joined. Proceed to step 6.

5.2 If this is not the first solution and $K < K'$, set $K' = K$, $L = 1$, and $S_{\tilde{i};\tilde{j}} = 0$ in the matrix of step 2.1. Note that (\tilde{i}, \tilde{j}) is the pair just suppressed and further, that (\tilde{i}, \tilde{j}) remains suppressed in all subsequent solutions. Go to step 5.4.

5.3 If $K \geq K'$ let $L \leftarrow L + 1$.

5.4 Maintain the routes formed and the order in which points were joined.

Step 6 - Suppress specified pairs

6.1 If $L < L'$, suppress the pair of points joined next in the current best solution, say (i', j') such that $S_{i;j'} = 0$ in the matrix of step 2.3 and return to step 3.

6.2 If $L = L'$ or if all pairs in the current solution have been suppressed, terminate at the current iteration. Proceed to step 7.

Step 7 - Different cities as a depot

7.1 If $s < |\mathcal{S}|$, let $s \leftarrow (s + 1)$. Take the next city from the set \mathcal{S} and return to step 1.5.

7.2 If $s = |\mathcal{S}|$, terminate the procedure and select the best solution from the entire group generated.

Sample Problems

To illustrate the above computational statement, a small Euclidean problem presented by Karg [22] will be used.

Table 2.1. Original Cost Matrix

$$C(1) = \begin{array}{c|ccccc} & 1 & 2 & 3 & 4 & 5 \\ \hline 1 & 0 & 30 & 26 & 50 & 40 \\ 2 & 30 & 0 & 24 & 40 & 50 \\ 3 & 26 & 24 & 0 & 24 & 26 \\ 4 & 50 & 40 & 24 & 0 & 30 \\ 5 & 40 & 50 & 26 & 30 & 0 \end{array}$$

Table 2.2. Cost Matrix Using City (2) as a Depot

$$C(2) = \begin{array}{c|ccccc} & 1 & 2 & 3 & 4 & 5 \\ \hline 1 & 0 & 30 & 24 & 40 & 50 \\ 2 & 30 & 0 & 26 & 50 & 40 \\ 3 & 24 & 26 & 0 & 24 & 26 \\ 4 & 40 & 50 & 24 & 0 & 30 \\ 5 & 50 & 40 & 26 & 30 & 0 \end{array}$$

Step 1:

The original cost matrix is given in Table 2.1. In this small problem $|\tilde{S}| = 5$. City (2) will be considered as a depot. $i^*, j^* = 2$. The new cost matrix to be used is depicted in Table 2.2.

Step 2:

From the cost matrix (Table 2.2), the saving matrix can be constructed as shown in Table 2.3. Note the computation of entry S_{4-5} for example: $S_{4-5} = C_{4;1} + C_{1;5} - C_{4-5} = 40 + 50 - 30 = 60$. All $S_{i;1}$ and $S_{1;j}$ are set at -1. The cost of the initial solution is simply the sum of the elements in the first row and the first column, in this case, both are the same, so it is only necessary to multiply the row (column) sum by 2. Hence, $K = 2 \times 144 = 288$.

Step 3:

The maximum element in S is found to be S_{4-5} with a savings of 60. It is important to mention that the algorithm searches through row by row the $\hat{S}_{i;j}$, and in Euclidean problems or in the case of tie, in general, the computer will take the first value found.

Step 4:

Since neither of the points are on a route, a route is constructed and the points are joined. The matrix saving is updated such that: $S_{4-5} = -1$; $S_{4;4} = 0$; and $S_{4-1} = S_{1-5} = 0$. The partial solution cost is computed as $288 - 60 = 228$.

Step 3:

Procedure returns to step 3 for a new $\hat{S}_{i;j}$. This time $S_{5;3} = 48$ is found to be the next maximum savings pair.

Step 4:

Point 5 is currently on a route, hence point 3 is assigned to the existing route. In the current saving matrix, set $S_{5;3} = -1$; $S_{3;5} = S_{5;1} = S_{1;3} = 0$. $K = 288 - 48 = 180$.

Step 3:

The next maximum saving corresponds to the joined pair (3-4), but in this case a subtour (3-4-5-3) is formed, therefore $S_{3;4}$ is set equal to zero, and a new candidate pair is considered. Finally, link (3-2) is placed into the solution and the final solution cost is: $K = 180 - 28 = 152$. Figure 2.4 illustrates the current final matrix solution and the order in which points were joined.

Table 2.3. Saving Matrix

	1	2	3	4	5
1	0	-1	-1	-1	-1
2	-1	0	28	20	40
3	-1	28	0	40	48
4	-1	20	40	0	60
5	-1	40	48	60	0

Table 2.4. Current Final Solution Matrix

	1	2	3	4	5		
1	0	0	0	-1	0	Order of Joined Pair	
2	-1	0	-	0	-		4-5
4	0	-1	0	-	-		5-3
4	0	-	-	0	-1		3-2
5	0	-	-1	0	0		

Step 5:

Since $K = 152 < 288$, the current solution is saved and K' is set at 152.

Step 6:

The suppression level L , is 1 and hence, the first pair selected

above is suppressed in the saving matrix, that is $S_{4;5}$ is set equal to zero, and the process is repeated with the new saving matrix. Suppression number 2 ($S_{5-3} = 0$) of the current solution $K = 152$ results in an improvement of $K = 148$. Consequently, the new best solution is saved and $K' \leftarrow K = 148$. Since a better solution has been found the pair (5-3) is permanently suppressed ($S_{5-3} = 0$). The new current solution is shown below in Table 2.5.

Table 2.5. Suppression Number 2 of $K = 152$

	1	2	3	4	5				
1	0	0	-1	0	0	Order of Joined Pair	Saving		
2	-1	0	0	-	-			4-5	60
3	0	-	0	-1	-			3-4	40
4	0	-	-	0	-1			5-2	40
5	0	-1	-	0	0				
								<hr/> Total Savings = 140	

Route: 1-3-4-5-2-1

Cost = $288 - 140 = 148$

Just as above, the first joined pair in the current best solution is removed, temporarily, from consideration such that $S_{4-5} = 0$ in the original saving matrix, ($S_{5;3}$ has been permanently removed). The problem is solved again by proceeding through step 3 and 4 such that a new solution is obtained with $K = 152$ which is discarded since $K' > 148$.

The next two points-pairs (3-4) and (5-2) are separately suppressed and solution of $K = 160$ are attained in both cases. Since all points-

pair have been exhausted interaction relative to city 1 terminates.

Step 8:

A new city is set as a central terminal (depot) and procedures return to step 1.5 to change the cost matrix and initialize a new solution iteration. The same optimal solution of 148 is obtained taking any city as a depot.

It was mentioned earlier that the presented algorithm can also treat non-symmetric problems. In order to show how it works, consider the problem presented by Conway [8].

Table 2.6. Cost Matrix

	1	2	3	4	5	6
1	0	1	7	3	14	2
2	3	0	6	9	1	24
3	6	14	0	3	7	3
4	2	3	5	0	9	11
5	15	7	11	2	0	4
6	20	5	13	4	18	0

Table 2.7. Saving Matrix

	1	2	3	4	5	6
1	0	-1	-1	-1	-1	-1
2	-1	0	4	0	16	0
3	-1	0	0	6	13	5
4	-1	0	4	0	7	0
5	-1	9	11	16	0	13
6	-1	16	14	19	16	0

Step 1 and 2:

Table 2.6 and 2.7 show the cost and saving matrix, respectively. City (1) is being considering the terminal. The sum of the first row and first column give an initial solution cost of 73.

Step 3:

Link (6-4) is selected as the first joined pair where $\hat{S}_{i;j} = 19$.

Step 4:

Points 6 and 4 are joined and so the first route is constructed. The partial solution cost is $73 - 19 = 54$. The saving matrix is updated such that $S_{6-4} = -1$; $S_{4-6} = 0$ and $S_{6;1} = S_{1;4} = 0$. Continuing this cycling process between step 3 and 4, the remaining joined pairs turn out to be (2-5), (5-6), and (4-3). Table 2-8 below shows the current final solution matrix along with the order in which pairs were joined.

Table 2.8. Current Final Solution Matrix

	1	2	3	4	5	6				
1	0	-1	0	0	0	0	Order of Joined Pair	Savings		
2	0	0	-	-	-1	-			6-4	19
3	-1	0	0	-	-	-			2-5	16
4	0	-	-1	0	-	0			5-6	13
5	0	0	-	-	0	-1			4-3	4
6	0	-	-	-1	-	0				
								Total Savings = 52		

$$\text{Cost} = 73 - \text{Total Savings}$$

$$73 - 52 = 21$$

Step 5:

Since $K = 21 < 73$, the current solution is maintained and K' is set equal to 21 and $L = 1$.

Step 6:

The first pair selected above is suppressed in the saving matrix

($S_{6:4} = 0$). Under the new saving matrix the process is repeated and final solution for suppression 1 of $K = 21$ is attained with a value of 20. Comparing with K' , the new best solution is maintained and $K' - K = 20$. Table 2.9 depicts the new best current solution.

Table 2.9. Final Solution Matrix for Suppression Number 1 of $K = 21$

	1	2	3	4	5	6				
1	0	0	-1	0	0	0	Order of Joined pair	Savings		
2	0	0	-	-	-1	-			2-5	16
3	0	-	0	-	-	-1			5-4	16
4	-1	-	0	0	-	-			6-2	16
5	0	0	-	-1	0	-			3-6	5
6	0	-1	-	-	-	0				
								Total Savings = 53		

Route: 1-3-6-2-5-4-1

$$\begin{aligned} \text{Soct} &= 73 - \text{Total Savings} \\ &= 73 - 53 = 20 \end{aligned}$$

Since a better solution has been found, the pair (6-4) is permanently suppressed. First link (2-5) of the new current solution is temporarily suppressed $S_{2:5} = 0$, and procedure returns to step 3 for the solution of suppression 1 of the current solution ($K = 20$).

Once the procedure has suppressed all joined pairs in the current solution, the final solution relative to city 1 is reached. Figure 2.2 presents a summary of the complete solution for city 1. Figure 2.3 shows the final solution cost for every city representing the

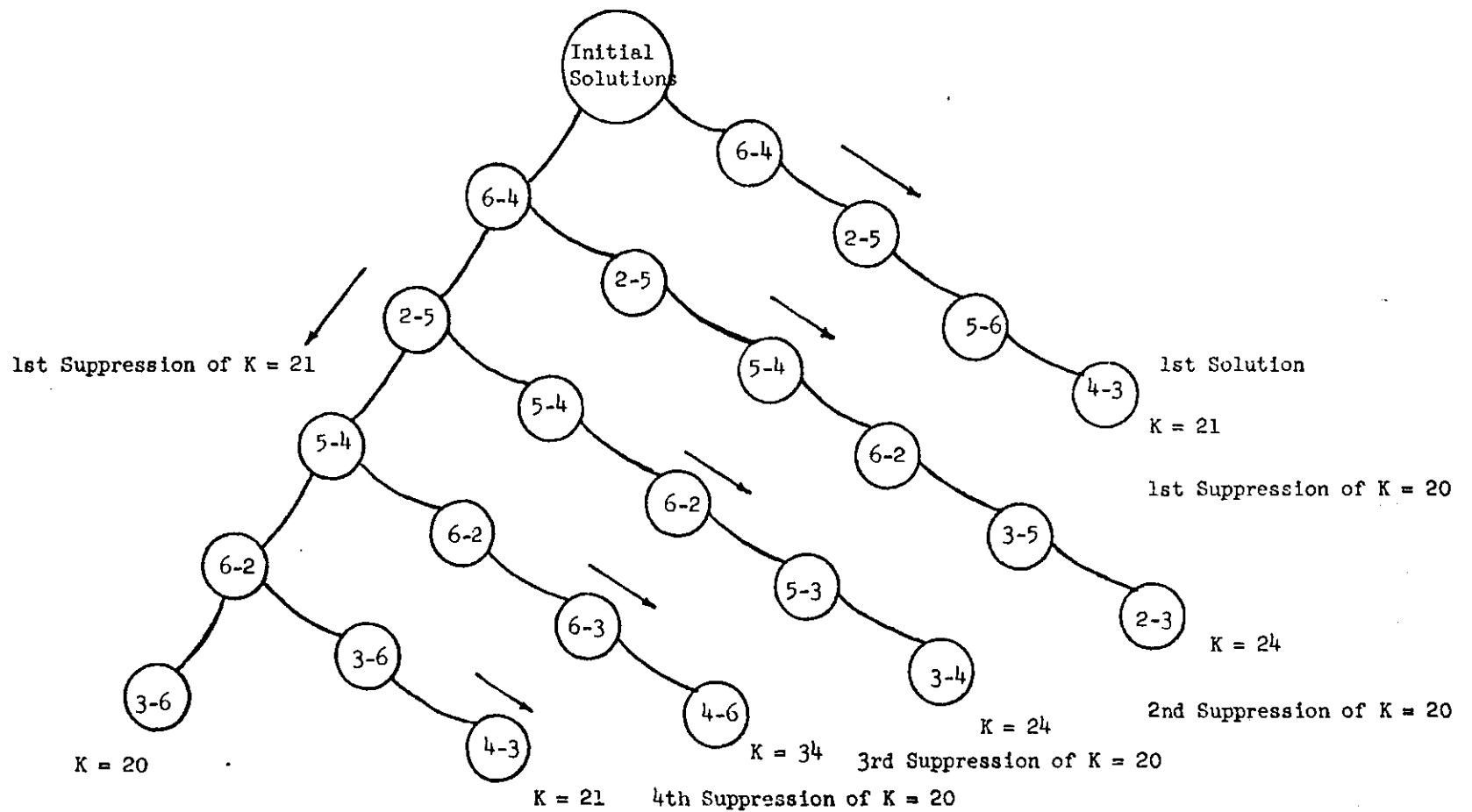


Figure 2.2. Summary of the Complete Solution for City (1) as a Depot

<u>City Taken as a Depot</u>	<u>Final Solution Cost</u>
1	20
2	20
3	20
4	21
5	20
6	20

Figure 2.3. Final Solution for Every City
Taken as a Depot

terminal.

Consider again the Euclidean problems. In order to show what the set \tilde{S} might look like for larger problems, consider Figure 2.4 in which a map is presented corresponding to the 42-city problem. The set \tilde{S} in this case can be considered [26, 25, 27, 24]. The best solution for this problem corresponds to that of taking city 26 as a depot - the nearest city to the centroid.

Computational Experience

A total of fifteen problems were solved to test the feasibility and efficiency of the proposed algorithm. These problems were taken from the literature so as to make possible comparison with results previously obtained. The quality of solution and computer times for all problems are summarized in Table 2.10. The quality of a solution obtained by the heuristic algorithm, referred to as efficiency, is defined as the ratio of the cost of the optimal (or best known) route and that obtained by the heuristic.

Out of the 15 problems solved in the experiment, optimal solutions known have been found for the first eleven problems. The optimal solutions are not available for problems 14 and 15. In problems 12, 13, 14 and 15, however, the efficiency of solutions were .98, .99, .99, .96, respectively.

For the problems with 25, 33, and 42 cities the best solutions were found using the 3rd, and 1st city closest to the centroid. In the first two and last problems respectively. In the 57 city-problem the best solution corresponded to a city far away from the centroid,

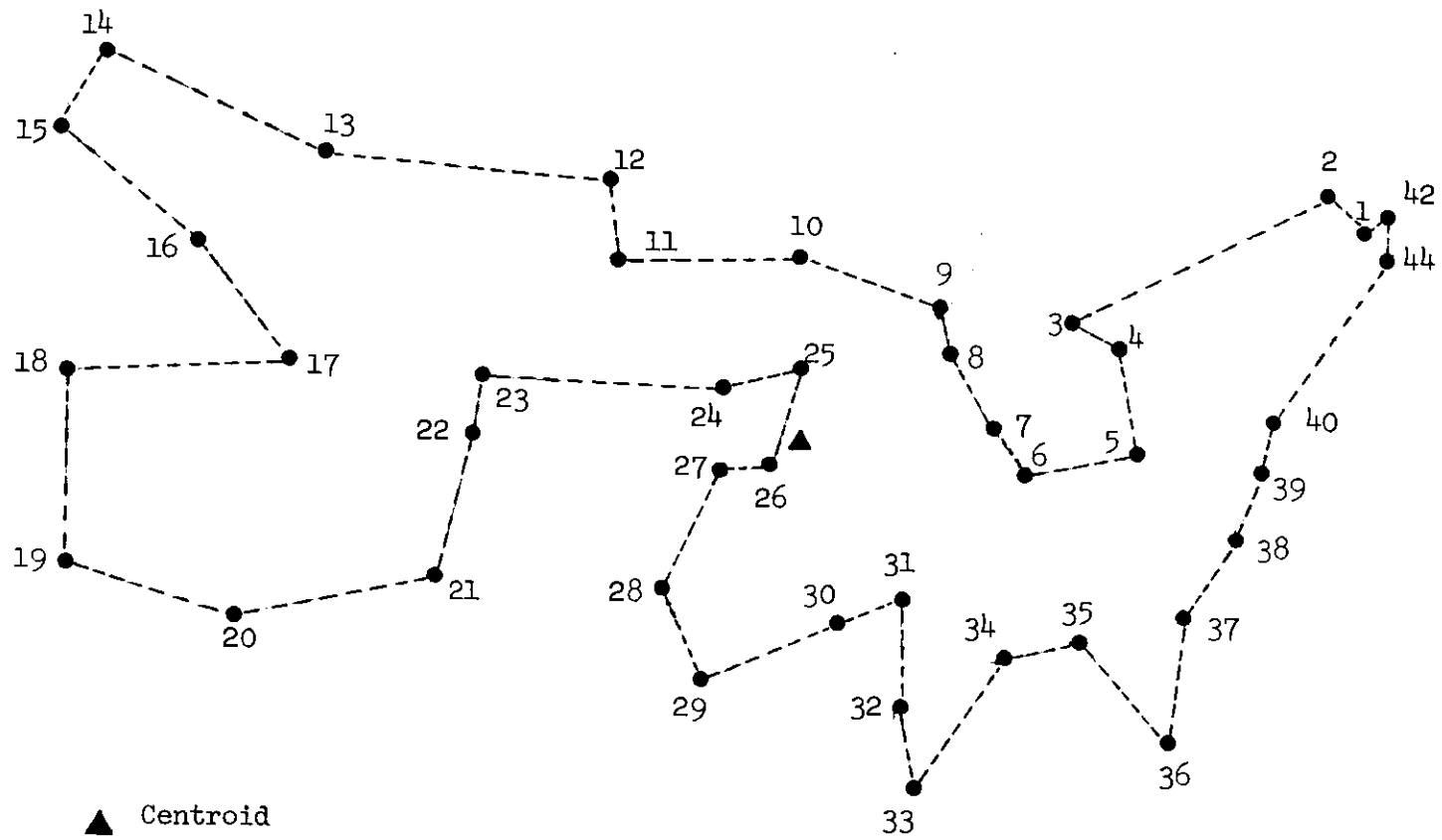


Figure 2.4. Best 42 City Tour Found

Table 2.10. Summary of Computation Experience. Comparative Result of Published Problem

Pro-blem #	n-City	Distance-Matrix Symmetric-Elements	Reference	Best Known Solution	Proposed Method		
					Solution	Efficiency	Time (Execution Time)
1	5	No - Not stated	Pierce (27)	20	20	1	Time Average Per Node: 0.314
2	5	No - Not stated	Wagner (32)	62	62	1	Run the Whole Problem: 1.57 sec.
3	5	Yes - Euclidean	Kar. & Th. (22)	148	148	1	Time Average Per City: 0.274
4	5	No - Not stated	Baker (2)	25	25	1	Run the Whole Problem: 1.64 sec.
5	6	Yes - Not stated	Cockran (7)	37	37	1	
6	6	No - Not stated	Little (24)	63	63	1	
7	6	No - Not stated	Conway (8)	20	20	1	The Average Per Node: 0.500
8	6	Yes - Not stated	Karp (20)	207	207	1	The Whole Problem: 5.00 sec.
9	6	No - Not stated	Montgomery (26)	21	21	1	
10	10	Yes - Euclidean	Karg & Th. (22)	378	378	1	
11	10	Yes - Euclidean	Ashour, Vega, (1)	3547	2547	1	
12	25	Yes - Euclidean	Held & Karp (20)	1711	1742	98%	4.32 Seconds Average Per Node
13	33	Yes - Euclidean	Karg & Thomp. (22)	10.861	10.894	99.7%	7.26 Seconds Average Per Node
14	42	Yes - Euclidean	Dantzing (11)	699	701	99.7%	15.36 Seconds Average Per Node
15	57	Yes - Euclidean	Karg & Thomp. (22)	12.955	13.209	98%	27.75 Seconds Average Per Node

* Problems were run on C.D.C. Cyber 74.

yet the second best solution is located at the 6th nearest to the centroid. It should be pointed out that an error might have been introduced by drawing the map, and the city considered the 6th might be the 4th or 5th since the points are very close in this area.

It is of interest to note that the range of solution efficiency taking the particular solution for any city as a depot is not overwhelming. The ranges are all about the same (approximately 10%) for the four large Euclidean problems. Table 2.11 indicates specific ranges.

Table 2.11. Range of Solutions

<u>Problem</u>	<u>n</u>	<u>Worst Solution</u>	<u>Best Solution</u>
12	25	.87	.98
13	33	.89	.99
14	42	.90	.99
15	57	.88	.98

Special attention should be given to the values shown in Table 2.11. Suppose for the 42-city problem, we take any city as a depot. The worst solution that might result will be that whose efficiency is 90%, with 15 seconds of computer time, which is considered a good solution. Even in problem number 12, an efficiency of 87% in 4 seconds results from random selection.

Execution times are given in Table 2.9 as average time per node. That is, the computer time that would be required to solve a problem taking any city as a terminal. Computational time increased

exponentially with the number of cities, and an equation can be constructed approximately such that

$$t = (e)^{.058n} \quad \text{where } t = \text{seconds and } n = \text{number of cities}$$

It is noted that the above formula is similar to that given by Svestka and Huckfeldt [31].

A very important conclusion from the experience with larger problems is that relative to the improvement obtained by the proposed algorithm over the original of Clarke and Wright. Improvements appear to decrease with the size of the problem. For the 25-city problem the improvement is 8% and decreases to 0% for the 57 city problem. This means that in this case the best solution is that given by Clarke and Wright.

CHAPTER III

THE DELIVERY PROBLEM

The Basis for an Approach

An algorithm is developed in this chapter which treats the classic vehicle delivery problem defined at the outset. Fundamentally, the procedure is structured around the notion of savings just as were the algorithms of Clarke and Wright and Holmes. However, as will be demonstrated, by employing a simple modification, a variation of both approaches can be constructed which can yield improved solutions. Prior to stating the proposed algorithm, however, it may be instructive to review the basic scheme of Clarke and Wright. A sample problem is used as a vehicle for demonstration.

Consider a small problem involving five demand points and a single terminal. The cost matrix is given in Table 3.1 as is the list of demands for all points. It is assumed that there is an unlimited number of 27-unit capacity vehicles. Notice, in this case the non-symmetry such that $c_{i,j} \neq c_{j,i}$. From the cost indices the saving matrix can be constructed as shown in Table 3.2. Note the computation of entry s_{3-5} for example: $s_{3;5} = c_{3;1} - c_{3;5}$; $s_{3-5} = 7 + 7 - 1 = 13$. An initial solution of one truck assigned to each demand point and is represented by an initial solution matrix with -1 in each cell of the first row and in each cell of the first column of S in Table 3.2 This means a truck is assigned from the central

The first step is to select the pair of points having the greatest savings as a candidate for joining. These points are tested against the constraints and joined if found feasible. For the given example, the pair (3-5) with a saving of 13 is selected first. The capacity constraint is met and the points are joined. The first step solution matrix is shown in Table 3.3. The joining of pair (3-5) is denoted by placing a -1 in cell (3-5) of the up dated solution matrix.

Table 3.3. The First Step Solution Matrix

	1	2	3	4	5	5
1	0	-1	-1	-1	0	-1
2	-1	0	11	9	-	8
3	0	-	0	-	-1	-
4	-1	4	8	0	-	4
5	-1	6	0	5	0	3
6	-1	0	9	6	-	0

This indicates that a truck is assigned to travel from point 3 to point 5. Note that now elements (3-1) and (3-1) and (1-5) are set equal to zero. This means that row 3 and column 5 need not be considered further for joining. The elements of row 3 represents all possible trips leaving demand point 3. Since each demand point is to be serviced, only one truck can leave any demand point. Therefore, once a trip from i is assigned, no more trips from i are possible. Notice also, that assigning a trip from i to j preclude the possibility of travel from j to i , since this would violate the constraint

that each point is visited only once. Therefore, whenever arc (i,j) is assigned, $s_{j;i}$ will be set equal to zero to avoid selection of (j,i) for joining.

Once row 3 and column 5 have been eliminated from further consideration, the procedure moves such that the next maximum saving is selected. The greatest savings at this step seem to be $s_{2;3}$ for the pair (2-3) on the same route so, $s_{2;3}$ is set equal to zero. By continuing in the same fashion, the next feasible joining corresponds to points (2-4). The corresponding reduced matrix is depicted in Table 3.4.

Table 3.4. Intermediate Solution Matrix

	1	2	3	4	5	6
1	0	-1	-1	0	0	-1
2	0	0	-	-1	-	-
3	0	-	0	-	-1	-
4	-1	0	8	0	-	4
5	-1	6	0	-	0	3
6	-1	0	9	-	-	0

Joinings are then made until no more are possible. From the intermediate solution matrix of Table 3.4, the last feasible-joined pair turns out to be (4-6). The final solution matrix is given in Table 3.5 as are the final routes and the cost of the final solution.

Hayes [18] identifies the two major faults with the above procedure. First, two customers once placed on a particular route

will remain together on the route in the final solution. Therefore, a link which initially appears advantageous may be the direct cause of a suboptimal solution. Second, demands are ignored in determining assignments of customers to route; except, of course, for the continual observation of the capacity constraints. Holmes [21] was led to the same conclusion, indicating that the selection of pairs for joining entirely on the basis of savings, may cause vehicles to be loaded less near capacity, or it may prohibit joining at later stages which would result in greater total savings.

Table 3.5. Final Solution Matrix

	1	2	3	4	5	6
1	0	-1	-1	0	0	0
2	0	0	-	-1	-	-
3	0	-	0	-	-1	-
4	0	0	-	0	-	-1
5	-1	0	0	-	0	-
6	-1	0	0	-	-	0

The final solution routes are:

Route 1	1-2-4-6-1	Savings	13
Route 2	1-3-5-1	Savings	13

Total Savings = 26

Final Solution Cost = 54 - Total Savings = 54 - 26 = 28

Based upon the above observations, it would seem desirable to formulate a different method for selecting an initial starting link. Suppose link (6-3) is selected as an initial joining. Table 3.6 shows the initial reduced saving matrix.

Table 3.6. The First Step Solution Matrix

	1	2	3	4	5	6
1	0	-1	0	-1	-1	-1
2	-1	0	-	9	11	8
3	-1	8	0	11	13	0
4	-1	4	-	0	5	4
5	-1	6	-	5	0	3
6	0	-	-1	-	-	0

Table 3.7. Intermediate Solution Matrix

	1	2	3	4	5	6
1	0	-1	0	-1	0	-1
2	0	0	-	-	-1	-
3	-1	8	0	11	-	0
4	-1	4	-	0	-	4
5	-1	0	-	5	0	3
6	0	-	-1	-	-	0

After deleting row 6 and column 3, we find that the highest savings correspond to points (3-5). However, the route that has been formed (6-3-5) is infeasible hence, $s_{3,5} = 0$. Returning, the next maximum

savings is found to be for the joining (2-5). Whenever there is a tie the algorithm selects the first element found. The final pair chosen is (3-4). The complete solution is shown below in Table 3.8.

Table 3.8. Final Solution Matrix Using Link (6-3) as Starting Point

	1	2	3	4	5	6
1	0	-1	0	0	0	-1
2	0	0	-	-	-1	-
3	0	-	0	-1	-	-
4	-1	0	-	0	-	0
5	-1	0	-	-	0	0
6	0	-	-1	-	-	0

The final solution routes are:

Route 1	1-2-5-1	Savings	11
Route 2	1-6-3-4-1	Savings	20

Total Savings = 31

Final Solution Cost = $54 - \text{Total Savings} = 54 - 31 = 23$

As can be seen, the solution has been improved by taking different link as the starting joining. This problem was solved by the suppression approach [21] (Holmes' procedure) and the final solution was also 23. The basic idea then is to find an appropriate starting link which tends not to prohibit subsequent lucrative joinings. A heuristic rule to find such a starting link has been

constructed which is based on an intuitive or heuristic notion which dictates that the optimal solution should contain, at least, the maximum savings of one of the demand points. In the next section a complete computational statement will be given. However, the two basic steps to be implemented in the basic scheme of Clarke and Wright are the following: First find, for each demand point, the link that yields the best savings. This is done by searching through the corresponding rows and columns of S . Then, rank those links in decreasing order of savings. These links will then be taken as starting joined pairs in the Clarke and Wright procedure. Note that the first link will correspond to that of the normal procedure.

There is one more comment which should be made on problems with different truck-capacities. Planning a system of routes for a fleet of trucks having different capacities poses a far more complicated structure than the single-capacity case. Clarke and Wright also allowed for this consideration, but as pointed out by Holmes, the vehicle capacities served only to act as a check after the pair had been selected for joining and the effects of capacity constraint upon optimality are not considered in the selection process. In the opinion of the author, one of the main achievements in this piece of work has been to find a suitable way to handle such cases.

Computational Algorithm

A step-by-step computational procedure of the proposed method will be given in this section. The method allows for the solution of symmetric as well as non-symmetric problems. In addition, equal or

different types of trucks are permissible.

Step 1 - Initialization

1.1 Construct an initial cost matrix C , such that $C = [c_{i;j}]$ $i, j = 1, 2, \dots, n$, where n is the number of demand points plus the terminal. When $i = j$, let $c_{i;j} = 0$.

1.2 Determine the demand of each point q_i , the number of trucks available of type k , T_k and the capacity of each \bar{c}_k .

1.3 Initialize the maximum saving link counter, IN at 1, and the greatest number of maximum savings links desired, IN' .

Step 2 - Compute the saving matrix and the initial solution.

2.1 Compute $s_{i;j}$ such that:

$$s_{i;j} = c_{i;1} + c_{1;j} - c_{i;j} \quad \forall i, j = 2, \dots, n, \text{ and } i \neq j$$

If $s_{i;j} < 0$ set $s_{i;j} = 0$. Set $s_{i;j} = 0$ for all $i = j$.

2.2 Set $s_{i;1} = -1$ for $i = 2, \dots, n$; $s_{1;j} = -1$ for $j = 2, \dots, n$.

2.3 Compute the cost of the initial solution such that

$$K = \sum_{i=2}^n c_{i;1} = \sum_{j=2}^n c_{1;j}$$

Step 3 - Find the starting links

3.1 Find the greatest savings $\hat{S}_{i,j}^{\hat{i},\hat{j}}$ corresponding to each demand point such that :

$$\hat{S}_{i,j}^{\hat{i},\hat{j}} = \overline{\text{Max}}_{(i,j)} [s_{i;j}] \quad \forall i, j = 2, \dots, n \text{ and } i \neq j$$

where $\overline{\text{Max}}_{(i,j)}$ is defined over all ordered pairs in row i and column

$j(i = j)$. If the problem is symmetric, the elements in the rank vector (starting links) can be selected in such a way that their opposites may be included or not. The two options are defined by the variable "simet" such that if $\text{simet} = 0$ $\hat{s}_{j;i}$ are set equal to zero, otherwise opposites can be included in the rank vector.

3.2 If $q_i^{\hat{i}} + q_j^{\hat{j}} > \bar{c}_1$ for $k = 1$ (the biggest truck), set $\hat{s}_{i;j} = 0$ (infeasible).

3.3 Rank the number of maximum saving links, IN' to explored, from the highest savings to the lowest. Go to step 5.

Step 4 - Determine a candidate pair

4.1 Find the ordered pair (i, j) with the greatest feasible savings such that,

$$\hat{S}_{i;j} = \underset{(i;j)}{\text{Max}} [s_{i;j}]$$

where (i,j) is defined over all ordered pair such that $s_{i;l}$ and $s_{l;j} \neq 0$.

4.2 If $\hat{s}_{i;j} = 0$ and trucks are equal, go to step 10, otherwise go to 7.2.

Step 5 - Join the points i and j on a route.

5.1 If this is the first joining set $\hat{i} = \hat{i}$ and $\hat{j} = \hat{j}$.

5.2 If neither of the points is on a route, construct a new route z and compute the required demand Q such that $Qz = q_i^{\hat{i}} + q_j^{\hat{j}}$. If equal trucks, go to step 6 otherwise go to 7.1.

5.3 If one of the points is currently assigned to a route, say z , attempt to join the unassigned point to z . Compute the total demand Q such that $Qz \leftarrow Q_z + q_i$ (or q_j) and if equal trucks go to step 6, otherwise go to 7.1.

5.4 If both points are currently assigned to routes say u and v , attempt to join both routes into one route, Z . Compute the total demand Q such that $Q_Z = Q_u + Q_v$. If equal trucks go to step 6 otherwise go to 7.1.

Step 6 - Attempt to assign the points to vehicles of equal capacity.

6.1 Check the capacity restriction such that; if $\bar{c}_1 \geq Q_Z$, proceed to step 6.2. If $\bar{c}_1 < Q_Z$, set $\hat{s}_i, \hat{j} = 0$ and return to step 4.

6.2 Update the number of trucks available T_1 such that $T_1 \leftarrow T_1 - 1$. If routes are joined increment the appropriate number of trucks available.

6.3 Compute the new solution cost such that $K \leftarrow K - \hat{s}_i, \hat{j}$, and return to step 4. Update the saving matrix such that $s_{i;j} = -1$; $s_{j;i} = 0$ and $s_{i;l} = s_{l;j} = 0$.

Step 7 - Attempt to assign the points to vehicles of different capacities

7.1 Each type of truck is considered separately, starting with the truck of largest capacity ($k = 1$). If $\bar{c}_k \geq Q_Z$ go to step 4. If $\bar{c}_k < Q_Z$ set $\hat{s}_i, \hat{j} = 0$ and return to step 4.

7.2 Consider all the routes formed under the truck size being considered.

7.3 If any of the routes can be assigned to a truck of smaller capacity, delete the route.

7.4 The savings of those routes that are only feasible for the truck being considered are calculated. Select among them the one(s) with highest saving.

7.5 As many trucks as available (1 per route) are assigned to the routes presenting highest savings.

7.6 Delete all routes not assigned to a truck.

Step 8 If the number of routes that have been formed under a type of truck (routes considered in 7.2) are less than the number of trucks available for that type, all demand points have been assigned to routes and therefore there is no need to consider further trucks. The algorithm assumes infinite number of trucks of the smallest capacity. Proceed to step 10.

If the number of routes is greater than the number of truck available consider the next type of truck and proceed to step 9.

Step 9 Consider the next type of truck

9.1 The truck(s) having the next largest capacity is considered such that $T_k \leftarrow T(k + 1)$. The savings matrix is reinitialized such that the points which are not on a route have their original values, and for the points already assigned to a route, $s_{i;j} = -1$, $s_{j;i} = 0$; $s_{l;j} = s_{i;l} = 0$. Return to step 4.

Step 10 - Save the best solution

10.1 If this is the first solution, save the cost K' such that $K' = K$ and go to step 11.

10.2 If this is not the first solution, and $K \leq K'$, set $K' = K$ and maintain the route formed.

Step 11 Check the termination conditions

11.1 If $IN < IN'$, take the next link in the maximum savings rank vector and return to step 5 with $IN \leftarrow IN + 1$.

11.2 If $IN = IN'$ or if all of the links in the rank vector have been considered, terminate the procedure.

Sample Problems

Prior to considering the application of the algorithm, two important comments should be made relative to the procedure. First, in order to gain insight into the method used to handle the problem of different capacities, consider the following example. Suppose we have two types of trucks "A" and "B", twelve demand points (including the terminal) and that we are in the "first pass" of the procedure. This means, the truck being considered is that of type "A". That is, the algorithm always start with the truck of maximum capacity. Further, assume that the information ultimately obtained is the following:

<u>Route Formed</u>	<u>Demand</u>	<u>Savings</u>	<u>Truck Considered</u>	<u>Next type of Truck</u>
Route 1 1-2-3-4-5-1	a	X	A	B
Route 2 1-6-7-8-9-1	b	Y	A	B
Route 3 1-10-11-12-1	c	Z	A	B

Assume also: $X > Y > Z$; ($a < A$ and $a < B$); ($b < A$ and $b < B$) and $c > B$.

Let the order in which joined pairs were selected be the following:

1. (2-3)
2. (6-7)
3. (11-12)
4. (3-4)
5. (7-8)
6. (10-11)
7. (8-9)
8. (4-5)

Step 7.3 of the algorithm requires the deletion of those routes that can be assigned to a truck of smaller capacity. In the hypothetical example, routes 1 and 2 are deleted. However, procedure makes sure that these routes will again result and will be feasible for the next smaller type of truck. This is warranted because the links were selected in decreasing order of savings. Hence, under truck type B the following routes are formed.

Route 1 1-2-3-4-5-1

Route 2 1-6-7-8-1

The new order in which pair were joined is:

1. (2-3)
2. (6-7)
3. (3-4)
4. (7-8)
5. (8-9)
6. (4-5)

Now both routes are assigned to truck of type B. The procedure then allows for a trade-off between demand and saving which contributes to better use of the truck capacities available.

A second point is concerned with the method of reduction relative to the saving matrix. In particular, the way in which pairs (j,i) given the initial ranking of some (i,j) , are handled should be given some thought. The example given in Table 3.9 can be used for clarification. Assume that each demand point has a demand of 2 and there is one truck of 8 unit-capacity and another of 4 units. Note that elements in the rank vector are different. Notice also that

the solution arising from the of 5-7. (1-5-7-2-6-1), (1-3-4-1) as starting joining is different from that when 7-5 is used. (1-7-5-2-6-1), (1-3-4-1). The difference in the rank elements, of course, produces a different solution. Computational results showing this effect are given in the next section.

Table 3.9. Saving Matrix for a Symmetric Example

	1	2	3	4	5	6	7		
1	0	-1	-1	-1	-1	-1	-1	Rank Vector when Taking the Opposite simet = 1	
2	-1	0	1	2	8	10	7		
3	-1	1	0	15	2	1	4		(5-7)
4	-1	2	15	0	2	3	4		(7-5)
5	-1	8	2	2	0	4	20		(3-4)
6	-1	10	1	3	4	0	4		(4-3)
7	-1	7	4	4	20	4	0		(2-6)
								(6-2)	
								Rank Vector without the Opposite simet = 0	
								(5-7)	
								(3-4)	
								(2-6)	

Consider now, two sample illustrations of the algorithm. The first deals with a problem when equal truck capacities exist and the second, when trucks are allowed to be different.

Case I Equal Trucks

The data for this problem is given below in Table 3.10. Note that point 1 is considered the depot and truck-capacity is 60 units. Steps 1 and 2 give the information summarized in Table 3.10(a) and 3.10(b). The cost K of the initial solution is the sum of the elements

in the first row and first column of C. That is $K = 34$.

Table 3.10(a) Cost Matrix

	1	2	3	4	5	6	7	
1	0	7	2	2	2	3	5	<u>Demand</u>
2	1	0	1	3	5	4	2	23
3	6	2	0	9	5	2	4	12
4	1	5	1	0	8	2	1	18
5	1	6	5	6	0	5	2	10
6	1	3	3	3	1	0	7	28
7	3	1	7	4	6	6	0	15

C =

Table 3.10(b). Saving Matrix

	1	2	3	4	5	6	7
1	0	-1	-1	-1	-1	-1	-1
2	-1	0	2	0	0	0	4
3	-1	11*	0	0	3	7*	7
4	-1	3	2	0	0	2	5*
5	-1	2	0	0	0	0	4*
6	-1	5	0	0	2	0	0
7	-1	9*	0	1	0	0	0

S =

The ordered pair that yield the best saving for each demand point are the following:

	<u>Link</u>	<u>Savings</u>
Point 2	(3-2)	11
Point 3	(3-2)	11
Point 4	(5-7)	5
Point 5	(5-7)	4
Point 6	(3-6)	7
Point 7	(7-2)	9

All links are seen to be feasible according to step 3.2, hence, step 3.3 yields the following rankings:

	<u>Link</u>	<u>Savings</u>
1 Max.	(3-2)	11
2 Max.	(7-2)	9
3 Max.	(3-6)	7
4 Max.	(4-7)	5
5 Max.	(5-7)	4

Assume that a maximum of five starting links are to be considered such that $IN' = 5$. The first element in the rank vector is taken as an initial starting joining. Table 3.11 shows the initial reduced savings matrix. At this point, the procedure moves to step 4 for the first $\hat{s}_{i;j}$ element which is $s_{4;7} = 5$. Neither of the points 4 and 7 are on a route so we construct a new one. The total demand for the resulting route is $18 + 15 = 33$ which is feasible. The two routes formed thus far are depicted in Table 3.12.

Table 3.11. Initial Reduced Matrix

	1	2	3	4	5	6	7
1	0	0	-1	-1	-1	-1	-1
2	-1	0	0	0	0	0	4
3	0	-1	0	-	-	-	-
4	-1	-	2	0	0	2	5
5	-1	-	0	0	0	0	4
6	-1	-	0	0	2	0	0
7	-1	-	0	1	0	0	0

Table 3.12. Intermediate Reduced Matrix

	1	2	3	4	5	6	7
1	0	0	-1	-1	-1	-1	0
2	-1	0	0	0	0	0	-
3	0	-1	0	-	-	-	-
4	0	-	-	0	-	-	-1
5	-1	-	0	0	0	0	-
6	-1	-	0	0	2	0	-
7	-1	-	0	0	0	0	0

From Table 3.12, it is seen that link (6-5) is the only feasible pair. Neither of the points is on a route, so a new one is formed. The total demand for this route is $28 + 10 = 38$, which is feasible for joining.

All remaining savings are zero hence a complete solution has been obtained with cost K computed as $34 - 18 = 16$. The cost K is better than the initial cost of 34 and the current solution is saved. The final routes and cost are depicted in Table 3.13.

Table 3.13. Final Solution Routes (Clark and Wright's Solution)

<u>Routes</u>	<u>Savings</u>
1-3-2-1	11
1-4-7-1	5
1-6-5-1	2

Total Savings = 18

Final Solution = Initial Cost - Total Savings

$$= 34 - 18 = 16$$

Relative to step 11, $IN \leftarrow IN + 1 = 2$. Thus the next link in the rank vector (7-2) is taken and set up as a first joined pair. The initial reduced matrix for the new link is depicted in Table 3.14. From step 4 pair (3-6) is the next candidate and the total demand is computed as $12 + 28 = 40$ which is feasible. Continuing, the next joined pair arises as (4-7). The total demand for the route (4-7-2) is $18 + 15 + 23 = 56$ which is feasible. The routes formed thus far are shown in Table 3.15.

Table 3.14. Initial Reduced Matrix for
the New Starting Link (7-2)

	1	2	3	4	5	6	7
1	0	0	-1	-1	-1	-1	-1
2	-1	0	2	0	0	0	0
3	-1	-	0	0	3	7	7
4	-1	-	2	0	0	2	5
5	-1	-	0	0	0	0	4
6	-1	-	0	0	2	0	0
7	0	-1	-	-	-	-	0

Table 3.15. Intermediate Solution Matrix

	1	2	3	4	5	6	7
1	0	0	-1	-1	-1	0	0
2	-1	0	2	0	0	-	-
3	0	-	0	-	-	-1	-
4	0	-	-	0	-	-	-1
5	-1	-	0	0	0	-	-
6	-1	-	0	0	2	0	-
7	0	-1	-	-	-	-	0

The next candidate pair corresponds to (2-3) where point 3 is currently assigned to a route. Checking for feasibility reveals that the demand $23 + 12 + 28 = 63$ exceeds the capacity available; hence $s_{2;3} = 0$. The final point arises as (6-5) after which the

algorithm returns to step 10 with a new cost of 11 which is better than 16 hence the new best solution is saved. Repetition of step 11 indicates that the solution given by the next starting link (4-7) is exactly the same as that for (7-2). In summary then, Figure 3.1 shows the different solutions which result from the different starting links. Table 3.16 shows the final solution of starting link (7-2).

Table 3.16. Final Solution Matrix for Link (7-2)

	1	2	3	4	5	6	7
1	0	0	-1	-1	0	0	0
2	-1	0	0	-	-	-	0
3	0	-	0	-	-	-1	-
4	0	-	-	0	-	-	-1
5	-1	-	0	-	0	-	-
6	0	-	0	-	-1	0	-
7	0	-1	-	-	-	-	0

Routes

Savings

1-4-7-2-1

14

1-3-6-5-1

9

Total Savings = 23

Final solution cost = 34 - 23 = 11

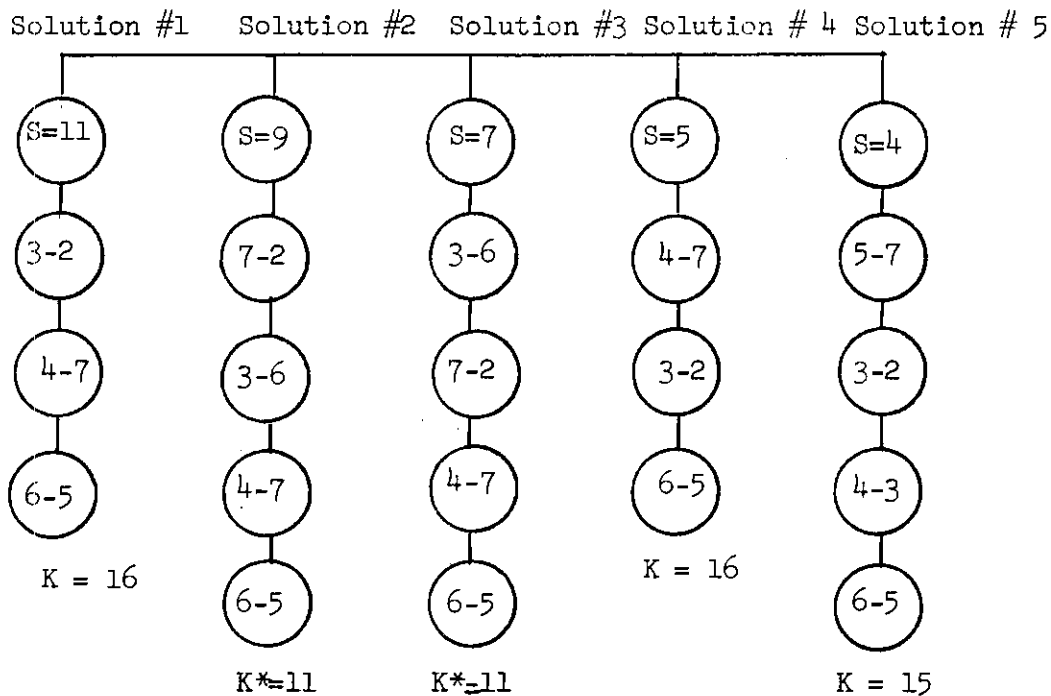


Figure 3.1. Summary of Different Solutions

Case II Different types of truck

An example given by Tillman and Cochran [32] will be used to demonstrate the method when the assumption of similar trucks is relaxed. All the required information is given in Table 3.17. Initializing: $IN^1 = 4$, $k = 1$ (start considering the routes when using the 18 units-capacity truck). Four different starting links are found and ranked as follows:

	<u>Link</u>	<u>Savings</u>
1 Max.	(5-7)	121
2 Max.	(4-6)	100
3 Max.	(3-7)	84
4 Max.	(2-4)	72

Table 3.17. Data for the Sample Case II

		1	2	3	4	5	6	7				1	2	3	4	5	6	7
C =	1	0	38	42	56	63	64	80	<u>Demand</u>	1	0	-1	-1	-1	-1	-1	-1	-1
	2	38	0	35	22	54	38	62	7	2	-1	0	45	72*	47	64	56	
	3	42	35	0	33	22	27	38	9	3	-1	45	0	65	83	79	84*	
	4	56	22	33	0	45	20	65	8	4	-1	72	65	0	74	100*	91	
	5	63	54	22	25	0	29	22	7	5	-1	47	83	74	0	98	121*	
	6	64	38	27	20	29	0	45	6	6	-1	64	79	100	98	0	99	
	7	80	62	38	65	22	45	0	5	7	-1	56	84	91	121	99	0	
									S =									

<u>Truck</u>	<u>Size</u>
1	18
1	15
∞	12

Once the link (5-7) has been set as the starting link, the procedure iterates by forming routes using the first type truck. The process is identical to that relative to case I. The result is displayed below and Table 3.18 showing the final solution matrix for the truck being considered. A more detailed explanation is given for steps 7, 8 and 9.

Step 7

<u>Routes</u>	<u>Demand</u>	<u>Savings</u>	<u>Truck Considered</u>	<u>Next type of Truck</u>
Route 1.1-5-7-1	12	121	18	15
Route 2.1-4-6-1	14	100	18	15
Route 3.1-2-3-1	16	45	18	15

The first two routes can be assigned to smaller trucks, so they are deleted. Route 3 is the only one left, hence the truck being considered is assigned to that route.

On step 8, the partial solution is computed as $686 - 45 = 641$.

Step 9

The next type of truck is considered (15-unit-capacity) and the initial saving matrix is reinitialized such that the points (2-3) are fixed on a route $s_{2;3} = -1$; $s_{3;2} = 0$, $s_{2;1} = s_{1;3} = 0$. The other points assume their original values. The new initial saving matrix for the truck type 2 is depicted in Table 3.19.

Table 3.18. Final Solution Matrix for Truck Type 1

	1	2	3	4	5	6	7
1	0	-1	0	-1	-1	0	0
2	0	0	-1	-	-	-	-
3	-1	0	0	0	0	-	-
4	0	-	-	0	-	-1	-
5	0	-	-	-	0	-	-1
6	-1	0	-	0	0	0	-
7	-1	0	-	0	0	-	0

Under the new type of truck the routes formed are:

<u>Routes</u>	<u>Demand</u>	<u>Savings</u>	<u>Truck considered</u>	<u>Next type of truck</u>
Route 1.1-5-7-1	12	121	15	12
Route 2.1-4-6-1	14	100	15	12

Table 3.19. Initial Saving Matrix for Truck Type 2

	1	2	3	4	5	6	7
1	0	-1	0	-1	-1	-1	-1
2	0	0	-1	-	-	-	-
3	-1	0	0	65	83	79	84
4	-1	72	-	0	74	100	91
5	-1	47	-	74	0	98	121
6	-1	64	-	100	98	0	99
7	-1	56	-	91	121	99	0

Step 7

Route 1 can be assigned to smaller truck, hence it is deleted and the truck being considered is assigned to route 2.

Step 8

The updated partial solution cost is $641 - 100 = 541$.

Step 9

The next truck type is considered, namely the 12-unit capacity-vehicle. The initial saving matrix is reinitialized such that points (2-3) and (4-6) are permanently assigned to a route.

Under the last type of truck the only route arising is 1-5-7-1 with a final solution cost of $541 - 121 = 420$. The algorithm assumes infinite numbers of truck of the lowest capacity, hence the procedure moves to step 10 and saves the new solution which is better than that obtained initially.

As in the case I, the procedure takes the second link in the

rank vector and goes through a new interaction. Table 3.20 shows the final solution taking link (2-4) as a starting joined pair. It is of interest to note that the solution is known to be optimal.

Table 3.20. Final Solution Matrix Taking Link (2-4) as Starting Joined Pair

	1	2	3	4	5	6	7
1	0	-1	-1	0	-1	0	0
2	0	0	-	-1	-	-	-
3	-1	0	0	-	0	-	-
4	-1	0	0	0	0	-	-
5	0	-	-	-	0	-	-1
6	-1	0	0	-	0	0	0
7	0	-	-	-	0	-1	0

<u>Routes</u>	<u>Savings</u>	<u>Truck</u>
1-5-7-6-1	220	18
1-2-4-1	72	15

Total Savings = 292

(Note that 1-3-1 is implied)

Optimal Solution Cost = Initial Cost - 292

$$= 686 - 292 = 394$$

Following, computational experience with the algorithm, for both cases, is discussed.

Computational Experience

Four types of problems were used to test the algorithm presented in the current chapter. The first class corresponds to symmetric and non-symmetric problems currently existing in the literature, three of which were different truck-sizes. This is done to provide an insight to the success of the modifications introduced to the Clarke and Wright procedure. The second problem is the multi-salesman case, which is a degenerate version of the truck delivery problem. The cost matrices for the 25, 33, 42 and 57 city problems used in Chapter II were utilized in the experimentation. The third type of problem is the general truck delivery. The same cost matrices above mentioned were used. Finally, a fourth class of problems with randomly generated, non-symmetric cost matrices were run.

In order to gain some basis of comparison in time and in quality of solution relative to the "suppression technique" of Holmes [21], all problems presented were also solved by latter approach.

Symmetric and Non-Symmetric Problems

The small problems existing in the literature were solved and the results are summarized in Table 3.21. Fundamentally, such problems are useful only for benchmark comparison since their sizes fall far short of those anticipated for "real" models.

Multi-Salesman Problem

As stated before, cost matrices for the 25, 33, 42 and 57 used in Chapter II were also utilized in this case. The capacities for the salesmen were assigned randomly. All the problems were solved

Table 3.21. Small Problems Existing in the Literature

Size n	Characteristics	Reference	Suppression		Proposed (Max. Saving)	
			Solution	Execution	Solution	Execution
6	Equal trucks - Symmetric	Hayes (18)	76	.3410*	76	.3470
7	3 types of trucks - Symmetric	Tillman (32)	394	.5810	394	.4420*
9	2 types of trucks - Symmetric	Holmes (21)	428	.7080	428	.4560*
6	Equal trucks - Not Symmetric	Parker (pag. 37)	23	.4050	23	.3470*
8	2 types of trucks - Not Symmetric	Holmes (21)	210	.4260*	210	.5070
7	Equal trucks - Not Symmetric	Parker (pag. 51)	11	.3600	11	.3400*

under the two alternatives; simet equal 0 and 1. Complete information for each problem is summarized in Table 3.22.

A very important comment on the solution found here and that given by the single-traveling salesman case should be made. Consider the 25-city problem with equal capacity-salesman. The optimal tour for the single-salesman using the same cost matrix is given by Raymond [30], and it is the following: 1-11-15-23-10-4-5-7-2-6-18-22-9-19-20-16-21-17-25-24-13-3-12-14-8-1. Cost - 1711. A suggested approach by Newton and Thomas [28] to solve the multisalesman problem, was to find the optimal tour for the single-salesman, and then "slice up" the route relative to the number of salesman that the problem dictates. If we do this for the example problem the results will be shown in Figure 3.2

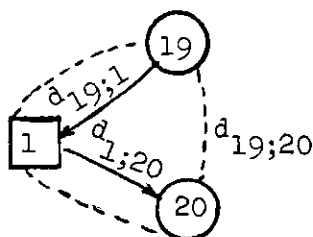


Figure 3.2. Solution of a Multi-Salesman Problem Based Upon a Single-Salesman Tour

$$\begin{aligned}
 \text{Solution Cost} &= \text{Optimal Cost for the Single-Salesman} + d_{19;1} + d_{1;20} - d_{19;20} \\
 &= 1711 + 201 + 183 - 18 \qquad d_{19;1} = 201 \\
 & \qquad \qquad \qquad d_{1;20} = 183 \\
 &= 2077 \qquad \qquad \qquad d_{19;20} = 18
 \end{aligned}$$

Table 3.22. Summary of Multi-Salesman Problems

Problem size, n	Truck-Number and Truck - Size			With Opposite						Without Opposite					
				Suppression		Proposed		Proposed		Suppression		Proposed		Proposed	
				L'=5	Time	IN'=15	Time	IN'=10	Time	L' =5	Time	IN'=15	Time	IN'=10	Time
25	∞	0	0												
	13	0	0	2076	5.63			2063	4.62	2082	6.27			1983*	5.45
	1	∞	0												
	20	3	0	1916	4.69			1916	5.02	1921	3.79			1907*	4.32
	11	8	3	2310*	7.68			2310*	6.59	2316	9.2			2310*	7.03
33	∞	0	0												
	15	0	0	14450	6.97	14408	11.86	14408	9.41	14450	6.24	14000*	11.81	14408	8.63
	1	∞	0												
	12	8	0	15391	11.63	15309*	19.45	15349	13.36	15391	10.32	15309*	19.38	15349	16.16
	12	10	5	15979	12.95	15802*	21.8	15839	16.63	15979	14.91	15802*	30.9	15839	17.3
42	∞	0	0												
	20	0	0	823*	20.6	830	25.9	830	16.2	926	16.11	827	23.6	830	18.00
	1	∞	0												
	15	12	0	1011	23.65	1009*	41.85	1009*	37.00	1011	25.41	1009*	46.31	1009*	26.4
57	1	1	∞												
	15	12	10	1064	21.16	1062*	58.6	1062*	43.3	1064	24.16	1062*	60.9	1062*	34.05
	∞	0	0												
57	21	0	0	14681	32.2	14681	68.67	14861	45.9	14681	26.5	14681	56.00	14681	46.74
	1	∞	0												
	20	15	0	15219	44.12	15219	112.1	15219	60.54	15219	40.3	15219	104.2	15219	66.1
	1	1	∞												
	20	15	10	15656	52.6			15636	71.5	15656	40.0			15656	83.3

The cost obtained is larger than the one given by the proposed method which has a cost of 1983 and the routes are as follows:

1-3-13-24-25-17-21-16-19-20-14-8-1

1-11-15-23-10-4-5-7-2-6-18-22-9-12-1

The quality of the solutions of the proposed algorithm are relatively "good". Table 3.23 shows the percentages of improvement obtained by proposed over the original of Clarke and Wright.

Table 3.23. Percentages of Improvement Over Clark and Wright Solutions

City-Problem	Equal Truck	2 Truck Type	3 Truck Type
25	10.67%	1.4%	2.65%
33	6.03%	3.4%	3.86%
42	10.69%	.19%	.18%
57	0.00%	0.00%	0.00%

It seems that the highest percentages of improvement occur when working with equal trucks such that the percentage decreases with changes in capacities. Regarding computational time, Figure 3.3 provides an idea, for the problems solved, of how the execution time fluctuates with the structure of the problem and the method used. The curves are constructed only for the case when opposites are considered (simet = 1). The times for the other cases are largely the same. The usual conclusion can be made from the plots such that the execution time increases with the size of the problem and

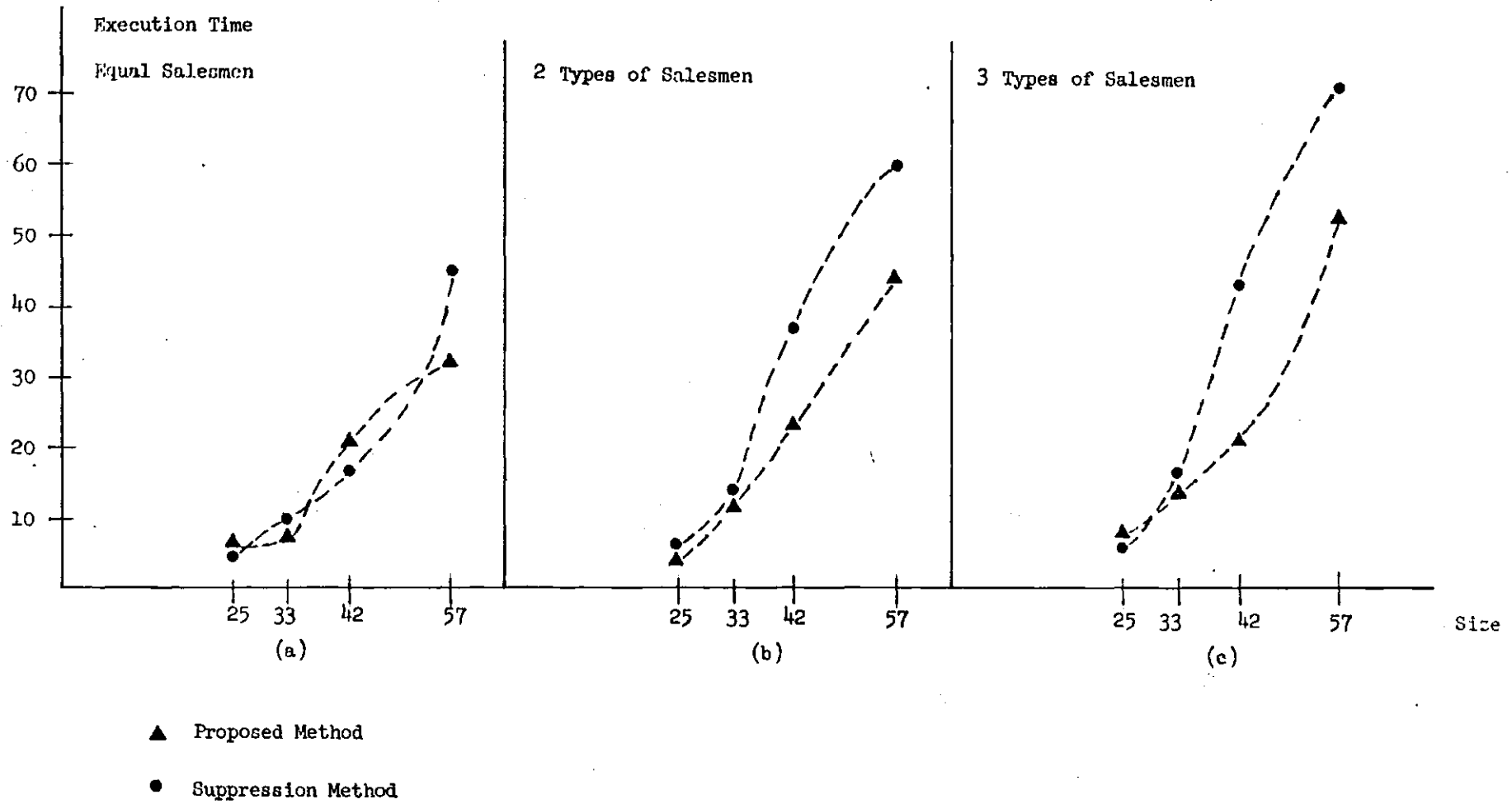


Figure 3.3. Computation Time of Multisalesman Problems

when working with different salesman-capacity.

Symmetric Truck Delivery Problem

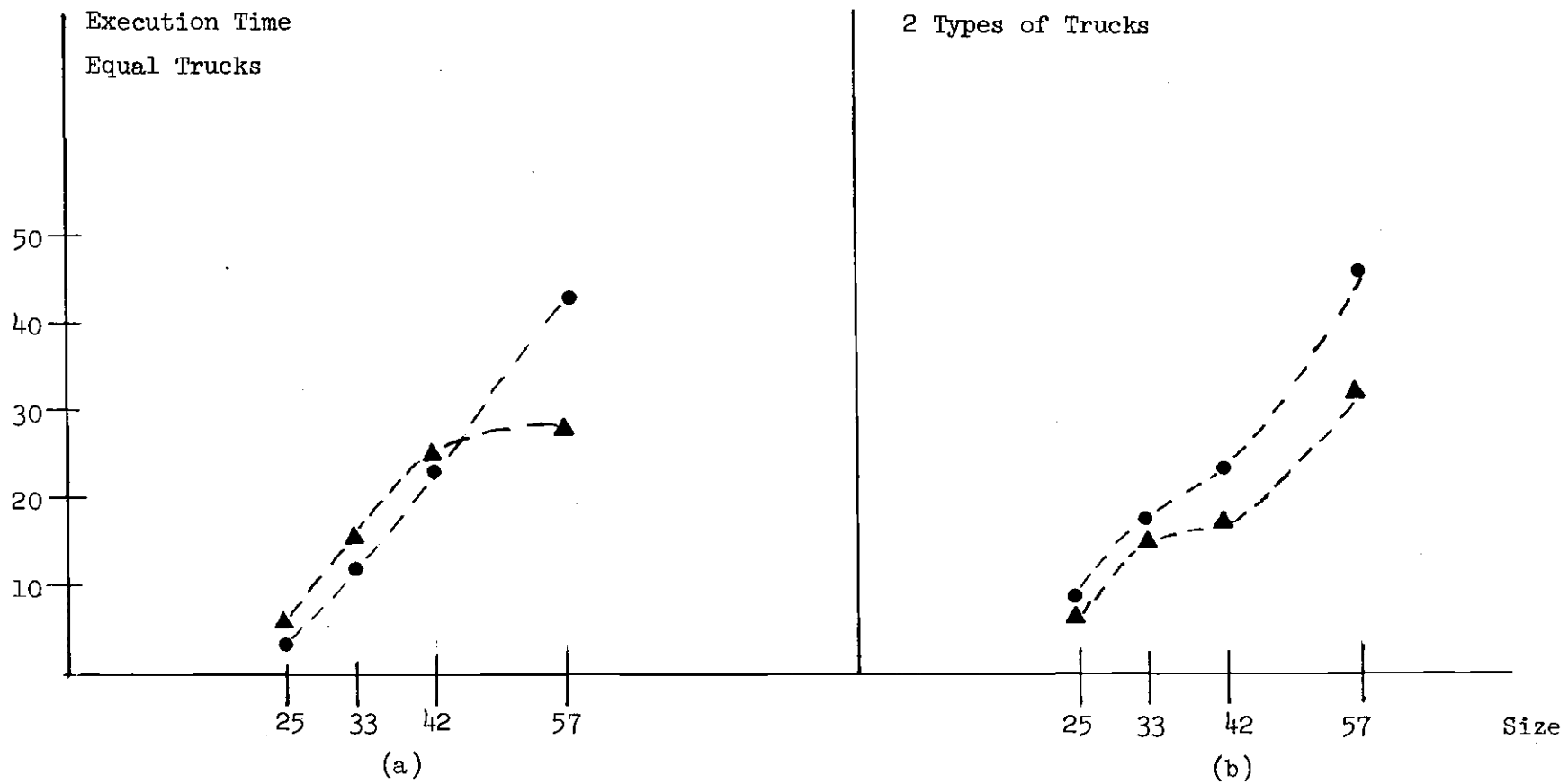
As previously indicated, the same cost matrices used in the experimentation of Chapter II are utilized here. The demand for each point and the capacity of trucks were assigned randomly. A complete compilation of the computational experience is given in Table 3.24. Figure 3.4 shows a rough behavior of time curves.

The highest percentage of improvement, 3.78 percent, relative to Clarke and Wright, was obtained in the 33-city problem when equal trucks were considered. Improvement for larger problems was more difficult to obtain. Hence, additional computation time involved would not normally be justified unless the attainment of solution improvement (or an attempted attainment) were of great importance and economically feasible. At any rate, from the experience obtained in the latter case and the one being considered, the following conclusions can be drawn.

1. Execution times are very sensitive to problem structure. That is, demands, number of trucks available for each size, truck-size, the cost matrix, and naturally the problem size, bear heavily upon the computational effort required.
2. Improvement for large problems is more difficult to obtain and further, such improvement decreases when different types of trucks are considered.
3. Another important factor determining the cost of solution and times involved appears to be the truck capacity. For a given

Table 3.24. Summary of Symmetric Truck Delivery Problems

Problem Size n	Truck-Number and Truck - Size		With Opposite				Without Opposite																							
			Suppression		Proposed		Suppression		Proposed																					
			L' = 5	Time	IN' = 10	Time	L' = 5	Time	IN' = 10	Time																				
25	∞	0																												
	120	0	2345*	6.00	2351	4.64	2351	5.73	2371	4.6																				
	2	∞																												
	100	50	2811	6.77	2811	8.13	2811	6.7	2761*	7.98																				
33	∞	0																												
	120	0	17597*	16.26	17661	12.29	17597*	14.46	17661	12.00																				
	2	∞																												
	100	50	20962	15.7	20920*	18.3	20962	13.8	20920*	18.08																				
42	∞	0																												
	150	0	1227*	25.49	1238	24.85	1227*	25.46	1238	24.13																				
	2	∞																												
	200	80	1107	15.4	1105*	23.28	1107	15.5	1105*	23.57																				
57	∞	0																												
	200	0	16330	28.23	16330	43.72	16330	28.21	16330	43.50																				
	2	∞																												
	250	100	16103	32.21	16103	46.63	16103	31.91	16103	46.46																				
Demand	0	16	15	12	13	12	10	16	11	14	18	12	18	12	15	14	12	11	16	19	20	15	14	15	10	12	14			
	16	15	16	13	13	11	18	15	13	12	14	16	12	10	10	12	12	10	11	14	14	16	13	12	13	16	12	12	10	10



- ▲ Proposed Method
- Suppression Method

Figure 3.4. Computation Time of Symmetric Truck Delivery Problems

problem, the effect of increasing truck capacity is that the number of trucks to be used is reduced. This implies that the number of points on a route increases and the size of the saving matrix is reduced much faster. Moreover, when more points are on a route, the savings tend to increase and the cost of the solution is reduced.

4. The proposed method appears to have an increasing computational time rate that is more or less constant. This is due to the fact that the algorithm is basically a repetition of the Clarke and Wright procedure. The suppression approach [21] establishes only a minimum number of suppressions, but the total number of suppressions depends on the structure of the problem. This is the reason for the curves' fluctuations in the suppression approach. It should also be mentioned that these conclusions are specified within the range of the problem sizes solved, and to extend this conclusion to larger problems would involve substantial speculation.

Randomly Generated Problems

A set of random cost problems were generated such that sizes of 20, 40, 60 and 80 points were constructed. Problems with 1 and 3 types of trucks were tested. A summary of the results are presented in Table 3.25. Similarly, Figure 3.5 indicates rough time curves.

Consider the following summary:

1. As in the symmetric case, execution time is affected by problem-size and truck type.

2. Improvement for the random non-symmetric case is more encouraging than for the symmetric case. This observation was also

Table 3.25. Summary of Randomly Generated Problems

Problem Size and Truck-Type	Suppression		Proposed	
	L' = 5	Time	IN' = 10	Time
20 Equal Trucks	208	2.11	206*	1.66
20 3 Types of Trucks	744	8.36	739*	8.71
40 Equal Trucks	313*	8.43	313*	11.24
40 3 Types of Trucks	641	37.4*	689	30.6
60 Equal Trucks	399	69.9	396*	33.32
60 3 Types of Trucks	685*	73.89	685*	74.7
80 Equal Trucks	486	61.98	486	88.5
80 3 Types of Trucks	700	121.35	698*	172.56

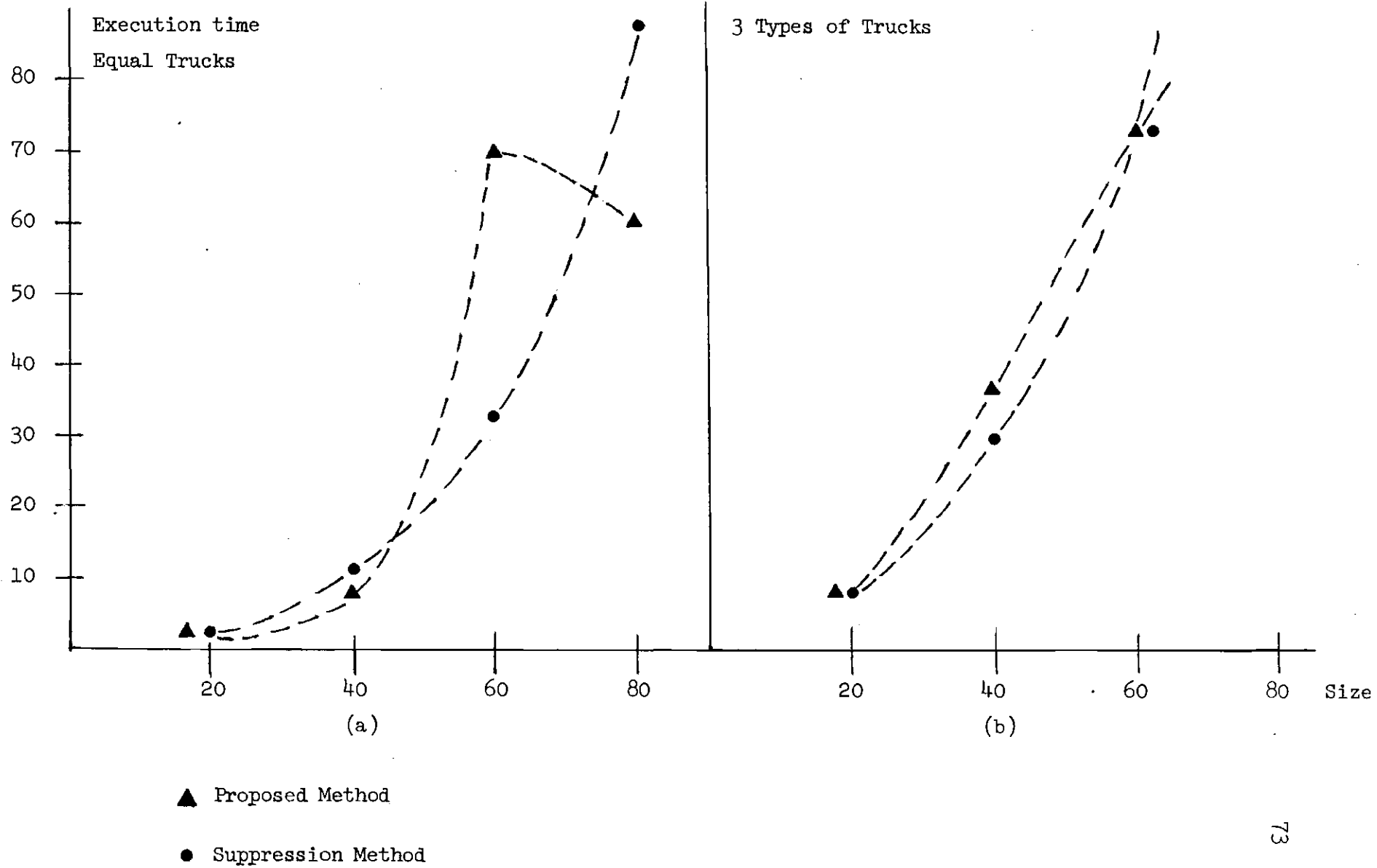


Figure 3.5. Computation Time of Non-Symmetric Randomly Generated Problems

made by Holmes such that the entries of the entries of non-symmetric cost matrix are completely unrelated, hence the solutions should show more variation.

3. It would appear that computationally, the non-symmetric problems require no more effort than do the symmetric problems.

CHAPTER IV

CONCLUSIONS AND EXTENSIONS

This research is concerned with the truck delivery and the traveling salesman type problems. The first problem is one of determining the "best" delivery routes, with respect to the distance traveled or other objectives, for a commodity where the system is subject to a set of constraints. In this problem, the delivery vehicles may have different capacities and all demands, which are assumed known, must be satisfied. Clearly the delivery problem may be considered a generalization of the classic traveling salesman problem which is one of finding the shortest route for a salesman starting from a given city, visiting each of a specified group of cities, and then returning to the starting city.

The general routing problem is important because the same problem structure may also be found in certain scheduling and sequencing problems other than that of routing vehicles. As an example, suppose that many different jobs had to be performed on a machine of a certain type, and the department involved had more than one such machine. For each job there is a set-up time which is dependent on the job just completed on the same machine. The problem then is to order the jobs on the machines so that the total set-up time is minimized.

Of the methods available for solving the delivery problem as well as traveling salesman problem (in fact most combinatorial

problems), the heuristic approaches appear to be the only ones currently feasible for solving large problems in a reasonable amount of time. With this in mind, two heuristic approaches were developed. Fundamentally, an algorithm presented previously was the basis for both approaches.

A heuristic approach for solving the delivery problem developed by Holmes [21], was modified to treat the traveling salesman problem. The structure of the latter problem implied the use of one of the given cities as a depot. Consequently, the notion of using a set of points near the centroid as candidates for a depot was developed. The heuristic performed well on small problems (symmetric and non-symmetric) such that optimal solutions to all such problems were obtained. For large sized problems ($n \geq 25$), only Eucliden problems were used and the solution quality was less than optimal but still of good quality. Hence, the results obtained from the application of the algorithm would appear to indicate its competitiveness, both in quality of solution as well as in computational effort. At any rate, a deeper investigation in the selection of the city to be used as a terminal, may lead to increased efficiency.

The algorithm presented to deal with the truck delivery problem is an extension of the Clarke and Wright [6] procedure. The modifications are applicable to both symmetric and non-symmetric problems. The basis for modification is that, as in the original method, using the link having maximum savings as an initial joining, may not be at all wise due to the truck capacity constraint. Consequently, different pairs of points were considered as starting links and the original procedure

was applied iteratively.

The algorithm allows the consideration of different truck-sizes (up to 5 types). In addition, the proposed algorithm performed well relative to known, optimal solution and also with reference to those which posed only "best-known" solutions. Also, the proposed vehicle delivery procedure was applied to multi-salesman traveling salesman problem. Results were encouraging.

For the delivery problem for symmetric cases, solution improvement was much more difficult to obtain. In general, the increase in computation time did not justify the solution improvement. In addition, the suppression approach seemed to perform better than the procedure suggested here in the equal capacity case. However, the situation is reversed when different types of trucks are considered.

Finally, improvements obtained from large, non-symmetric problems (random) were more encouraging than those from the symmetric case. In terms of quality of solution, the suggested method produced slightly better solutions than that in [21].

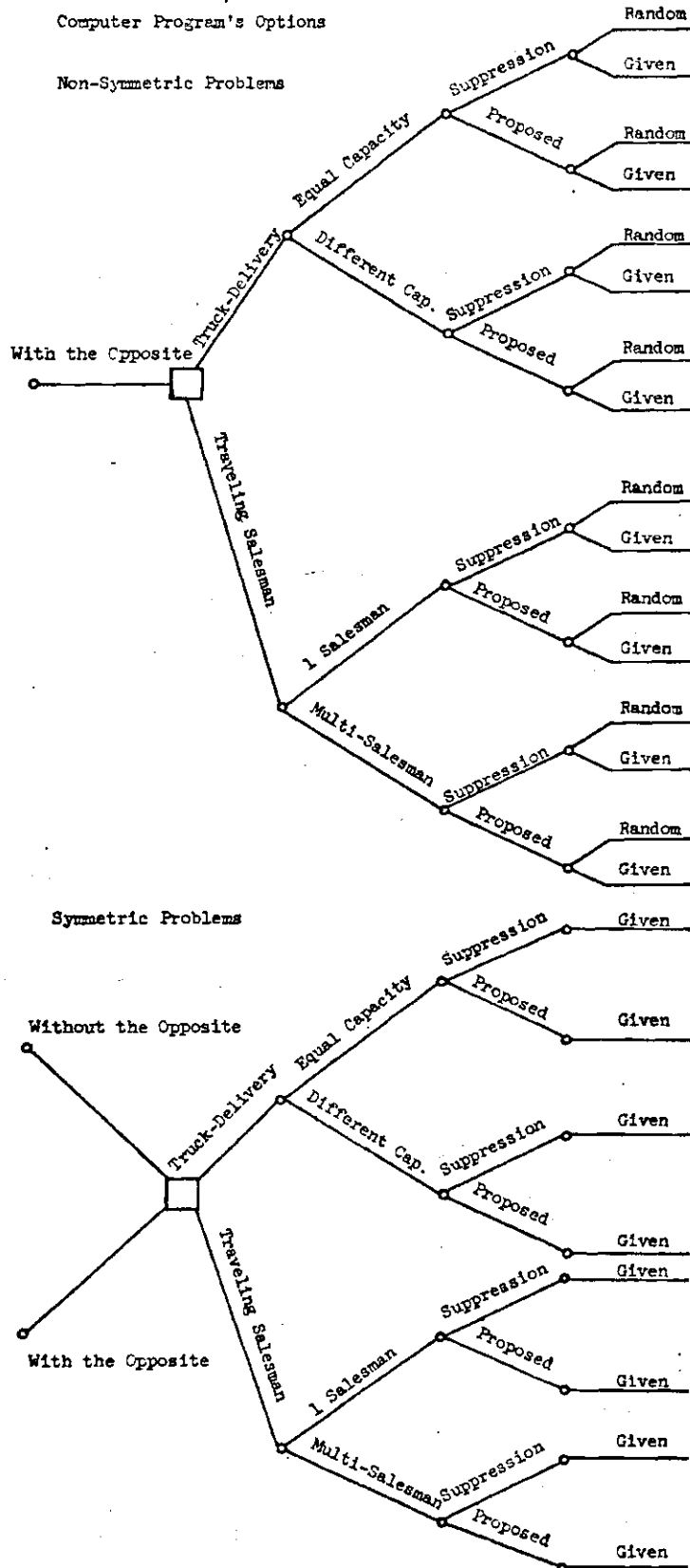
On the whole the results obtained reflect the feasibility and efficiency of the proposed work. However, there would seem to areas for improvement and extensions. An analysis of greater discretion in the selection of the initially joined pair as well as in the number of such initial joinings to be investigated may be a subject of further studies. Criteria such as using elements in the row and column belonging to the maximum saving link, links having a large saving to demand ratio and joinings which after deletion of the respective row and column leave maximum overall savings, should be investigated.

The notion of finding an appropriated initial joining to the Clarke and Wright procedure, may suggest that under certain conditions of demand and truck capacity, the saving approach can be converted in an exact technique. For example, the application of both suppression and initial link techniques together, may result in a branch-and-bound scheme. This would seem to be a worthy area on which to focus additional work.

APPENDIX

COMPUTER PROGRAM'S OPTIONS, DATA

INPUT AND LISTING



DATA INPUT

Size

IXRD: 1 No Random (Given)
 0 Random

ISIMET 1 With the Opposite
 0 Without the Opposite

Supp 1 Suppression
 0 Proposed

ICON n Proposed (number of points in the Rank Vector)
 0 Suppression

TRKDEL 1 Truck Delivery
 0 Traveling Salesman

NUM TRUCK

DEMAND

COST MATRIX

CAPACITY

* Demand, Cost Matrix, and Capacity are generated in random problems.

PROGRAM MAIN(INPUT,OUTPUT,TAPE5=INPUT,TAPE6=OUTPUT)

THIS MODULE IS DESIGNED AS A HEURISTIC SOLUTION TO THE
VEHICLE SCHEDULING PROBLEM. THE MODULE SOLVES NONSYMMETRIC
PROBLEMS HAVING UP TO 95 NODES. THE COST MATRIX IS
RANDOMLY GENERATED

THIS PROGRAM WAS DESIGNED BY IVAN J PEREZ DURING
THE FALL QUARTER OF 1975

DIMENSION DIST(95,95,4),DEMAND(95),CAPCTY(5),NUMTRK(5)
* DIMENSION ROUTE(40,23,2),RANK(100,2,2),X(101)
INTEGER SYM,SUPP,SIZE,DIST,DEMAND,CAPCTY,RANK,ROUTE
* ,C(95,95)
LOGICAL SW,SWITCH
COMMON //SIZE,JPROD,DEMAND,CAPCTY,SUPP,NUMTRK,IXRO
* /BLOCK1/DIST,ROUTE,KK,K,L,C,SWITCH
* /BLOCK2/ICOST,JCOST,NEWCST
* /BLOCK3/ICON,SW,RANK,ICUAL,ICONT
* /BLOCK4/JOINR,JOINC
* /BLOCK5/ITRDEL
* /BLOCK6/ISIMET

EXPLANATION OF VARIABLES

DIST - MATRIX TO STORE THE FOLLOWING
1 - DISTANCE OR COST MATRIX
2,3,4 - SAVINGS MATRIX FOR THE SOLUTION PROCEDURES
ROUTE - ROUTES GENERATED BY THE PROGRAM
ISIMET - SUPPRESSION OF SIMETRIC POINTS IS DESIRED IF THE
VALUE OF THIS VARIABLE IS 0
ITRDEL - CONTROLS THE TRAVELING SALESMAN PROBLEM OR THE
TRUCK DELIVERY PROBLEM, IF THE VALUE OF THIS
VARIABLE IS 1 A TRUCK DELIVERY IS EXECUTED
SIZE - GIVES THE DIMENSION OF THE COST MATRIX
DEMAND - GIVES THE DEMAND OF EACH NODE
CAPCTY - GIVES THE CAPACITY OF THE TRUCKS USED, FIVE TYPES
ARE ALLOWED
NUMTRK - GIVES THE NUMBER OF TRUCKS AVAILABLE FOR EACH TYPE
ICOST - STORES THE COST OF THE BEST SOLUTION
JCOST - STORES THE COST OF THE STARTING SOLUTION
NEWCST - STORES THE COST OF THE SOLUTION BEING GENERATED
SWITCH - CONTROLS THE GENERATION OF THE RANDOM MATRIX
SW - CONTROLS THE EXECUTION BY THE METHOD OF MAXIMUM SAVINGS
IXPD - SPECIFIES WHETHER A RANDOM MATRIX IS TO BE GENERATED
RANK - USED TO STORE THE ORDER IN WHICH THE POINTS ARE SELECTED
KK - SPECIFIES WHICH ROUTE MATRIX IS BEING USED
KJ - SPECIFIES WHICH RANK MATRIX IS BEING USED
L - SPECIFIES WHICH TRUCK TYPE IS BEING USED
MAXSAV - SPECIFIES THE VALUE OF THE SAVINGS OF THE POINT
SELECTED FOR INTRODUCTION IN THE ROUTE
JOINR - ROW OF THE POINT SELECTED
JOINC - COLUMN OF THE POINT SELECTED
SUPP - SPECIFIES WHETHER THE METHOD OF SUPPRESSIONS SHOULD
BE USED OR NOT, USED IF IT HAS THE VALUE 1
ICON - SPECIFIES THE NUMBER OF POINTS DESIRED FOR CONSIDERATION
WHEN USING THE METHOD OF MAXIMUM SAVINGS
ICUAL - SPECIFIES WHICH MAXIMUM SAVINGS IS BEING TAKEN TO
GENERATE THE SOLUTION
ICONT - SPECIFIES THE NUMBER OF MAXIMUM SAVINGS THAT HAVE BEEN
GENERATED WITH THE METHOD OF MAXIMUM SAVINGS
JPROD - USED TO CONTROL THE POTATION OF THE DISTANCE MATRIX
WHICH ALLOWS TO STUDY EVERY NODE AS THE TERMINAL

```

C      IPROM - PERFORMS THE SAME FUNCTION AS JPROR
C      X - VECTOR USED TO STORE RANDOM NUMBERS BETWEEN 0 - 1.
C      K - SPECIFIES WHICH SAVINGS MATRIX IS BEING USED
C      C - MATRIX TO KEEP AN UNMODIFIED VERSION OF THE COST MATRIX
C      TIME - VARIABLE USED TO PRINT SEVERAL TIMES THROUGHOUT THE
C      EXECUTION
C      ICAPCK - STORES THE SUM OF THE CURRENT USED CAPACITY PLUS
C      THE DEMAND OF THE POINTS TO BE JOINED
C      MARK - VARIABLE USED TO INDICATE HOW MANY LINES OF THE
C      ROUTE MATRIX HAVE BEEN USED. SO NOTHING IS WRITTEN
C      OVER THEM WHEN USING THE MAXIMUM SAVINGS METHOD
C      COST - VECTOR USED TO CALCULATE THE SAVINGS OF EACH ROUTE
C      GENERATED. THIS IS DONE WHEN THE PROGRAM RUNS WITH
C      MORE THAN ONE TYPE OF TRUCK

```

```

C-----
C      TIME=SECOND(0.)
C      TIMEST=TIME0

```

```

C
C      TAKE THE CURRENT TIME AS OF THE START OF THE PROGRAM IN ORDER TO
C      DETERMINE THE EXECUTION TIME

```

```

C
C      SWITCH=.FALSE.
C      READ(5,*)SIZE
C      READ(5,*)IXRD,ISIMET

```

```

C
C      READ IN THE SIZE OF THE COST MATRIX, WHETHER IT SHOULD
C      BE RANDOMLY GENERATED OR NOT, AND WHETHER SUPPRESSION OF
C      SYMMETRIC POINTS IS DESIRED

```

```

C      THE FOLLOWING IS THE INITIALIZATION STAGE
C-----

```

```

C      XTRK=0
C      IPTRB=0
C      IPROM=SIZE
C      JPROR=0
3617 CONTINUE
C      CALL IVAN.RETURNS(3618)
C      ICOST=0
C      DO 6464 I=1,40
C      DO 6465 J=1,23
C      ROUTE(I,J,1)=0
C      ROUTE(I,J,2)=0
6465 CONTINUE
6464 CONTINUE
C      DO 7464 I=1,100
C      DO 7465 J=1,2
C      RANK(I,J,1)=0
C      RANK(I,J,2)=0
7465 CONTINUE
7464 CONTINUE

```

```

C-----
C      WRITE THE DATA TAKEN AS INPUT
C-----

```

```

C      WRITE(6,1319)
C      WRITE(6,1320)
C      WRITE(6,1321)SIZE
C      IF(NUMTRK(2).EQ.0) GO TO 4242
C      WRITE(6,1325)(NUMTRK(J),J=1,5)
C      GO TO 5353
4242 WRITE(6,7574) NUMTRK(1)

```

7574 FORMAT(///I4,4#H MACHINE "PROBLEM" ALL MACHINES HAVE = CAPACITY)
 5353 C O N T I N U E

```

-----
C      IF(IXP0.NE.0.OR.SWITCH)GO TO 4736
C
C      THIS IS THE CASE WHEN EITHER NO RANDOM MATRIX IS DESIRED
C      OR IT HAS ALREADY BEEN GENERATED
C
C      GENERATION OF THE RANDOM COST MATRIX, DEMAND, AND
C      TRUCK CAPACITY
-----
      READ(5,*)IU
      DO 59 I=1,SIZE
      CALL RAND(IU,SIZE,X)
      DO 58 J=1,SIZE
58     DIST(I,J,1)=X(J)*100.
      DIST(I,I,1)=0
59     C O N T I N U E
      CALL RAND(IU,SIZE,X)
2002  FORMAT(25I4)
      IF(IPTRO.EQ.1) GO TO 9735
C
C      THIS PORTION PERTURBS THE COST MATRIX SUCH THAT POSITIVE SAVINGS
C      RESULT. FOR THE TRAVELING SALESMAN PROBLEM
C
      DO 3422 I=1,SIZE
      DO 3422 J=1,SIZE
3422  C(I,J)=DIST(I,J,1)
      DO 62 I=1,SIZE
62     WRITE(6,2002)(DIST(I,J,1),J=1,SIZE)
      IF(ITPDEL.EQ.1)GO TO 8735
      DO 4722 J=2,SIZE
      DIST(I,J,1)=DIST(I,J,1)+500
8732  CONTINUE
      DO 4733 I=2,SIZE
      DIST(I,1,1)=DIST(I,1,1)+500
8733  CONTINUE
C
8735  CONTINUE
C
C      ICOST=0
      CALL RAND(IU,SIZE,X)
      RHO=0
      NIZ11=SIZE-1
      DO 7879 I=1,NIZ11
      DEMAND(I+1)=X(I)*10.0+10.0
      IF(ITPDEL.NE.1)DEMAND(I+1)=1
7879  RHO=RHO+DEMAND(I+1)
      DO 7010 ILX=2,5
      IF(NUMTRK(ILX).EQ.0)GO TO 3020
3010  CONTINUE
      ILX=6
3020  ILX=ILX-1
      DO 3030 IL1=1,ILX
3030  XTRK=XTRK+NUMTRK(IL1)
      DO 3040 IL1=1,ILX
3040  CAPCTY(IL1)=(RHO*NUMTRK(ILX-IL1+1)/XTRK)/2.
      IF(NUMTRK(2).NE.0)GO TO 3050
      IMM=RH0/XTRK
      CAPCTY(1)=IMM+IMM*0.30
3050  WRITE(6,1722)
6320  FORMAT(/F10.1///)

```

```

WRITE(6,1304)(DEMAND(J),J=1,SIZE)
3420 FORMAT(////45H THE IDEAL MACHINE LOADING IS LISTED BELOW )
WRITE(6,3420)
WRITE(6,1323)
WRITE(6,1324)(CAPCTY(J),J=1,5)
SWITCH=.TRUE.
GO TO 100
8736 CONTINUE
C
C-----
C INITIALIZE THE SAVINGS TO ONE TRUCK FOR EACH NODE
C-----
WRITE(6,1422)
1422 FORMAT(* THE DEMAND AT EACH OF THE POINTS, N=2,...,N, ISZ*)
WRITE(6,1409)(DEMAND(J),J=1,SIZE)
1408 FORMAT(25I3)
WRITE(6,1423)
1423 FORMAT(/** THE SIZE AND NUMBER OF TRUCKS AVAILABLE AREZ*)
WRITE(6,1424)(CAPCTY(J),J=1,5)
1424 FORMAT(* TRUCK SIZE *,7X,5I5)
100 DO 110 I=2,SIZE
ICOST=ICOST+DIST(I,1,1)+DIST(1,I,1)
DO 110 KS=2,4
DIST(I,1,KS)=-1
110 DIST(1,I,KS)=-1
JCOST=ICOST
C
C-----
C CALCULATE ALL SAVINGS AND STORE THEM IN THREE SEPERATE
C MATRICES ONE FOR THE INITIAL SOLUTION, ONE FOR THE
C WORK AREA, AND ONE FOR THE CURRENT BEST SOLUTION.
C-----
DO 120 I=2,SIZE
DO 120 J=2,SIZE
IF(I.EQ,J)GO TO 120
DIST(I,J,2)=DIST(1,1,1)+DIST(1,J,1)-DIST(1,J,1)
DIST(I,J,2)=MAX0(DIST(I,J,2),0)
DIST(1,J,2)=DIST(I,J,2)
120 DIST(I,J,4)=DIST(1,J,2)
C-----
C WRITE THE SAVINGS MATRIX AS HAS BEEN GENERATED
C-----
WRITE(6,1327)
DO 1199 I=1,SIZE
1199 WRITE(6,1303)(DIST(I,J,2),J=1,SIZE)
JCOST2=JCOST
IF(I.PEQL.NE.1)JCOST2=JCOST-((SIZE-1)*1000)
WRITE(6,1324)JCOST2
C-----
C THE PROGRAM STARTS ASSIGNING ROUTES
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
126 CALL MARIA,RETURNS(181,2100)
C INITIALIZE VARIABLES FOR THE FIRST RUN
C-----
I4=1
K=3
KK=1
K3=1
L=1
127 NEWCST=JCOST

```



```

-----
C      MOVE ALL THE POINTS TO ALLOW THE NEW POINT TO ENTER
-----
C
350 IF(ROUTE(II,19,KK).EQ.0)GO TO 352
    IF(ROUTE(ROUTE(II,19,KK),18,KK).EQ.99999)GO TO 400
    IF(ROUTE(ROUTE(II,19,KK),18,KK).EQ.0)GO TO 400
353 ROUTE(II,JJ,KK)=ROUTE(II,JJ-1,KK)
    JJ=JJ-1
    IF(JJ.NE.1)GO TO 353
370 IF(ROUTE(II,19,KK).EQ.0)GO TO 380
    ROUTE(II,1,KK)=ROUTE(ROUTE(II,19,KK),18,KK)
400 II=ROUTE(II,19,KK)
    JJ=18
    GO TO 350
380 ROUTE(II,1,KK)=I
    ICAPCK=OMAND(II)+ROUTE(II,21,KK)
390 ROUTE(II,21,KK)=ICAPCK
    II=ROUTE(II,20,KK)
    IF(II.EQ.0)GO TO 237
    GO TO 300
352 JJ=18
    GO TO 353
-----

```

```

-----
C      ATTEMPT TO JOIN TWO EXISTING ROUTES
-----
C
600 II=1
620 IF(ROUTE(II,19,KK).EQ.0)GO TO 610
630 II=II+1
    IF(II.GT.40)GO TO 1900
    GO TO 620
610 IF(ROUTE(II,1,KK).EQ.0)JOINC)GO TO 640
    GO TO 630
640 ISAV=II
645 IF(ROUTE(II,20,KK).NE.0)GO TO 650
    JJ=1
660 IF(ROUTE(II,JJ,KK).EQ.0)JOINR)GO TO 740
    JJ=JJ+1
    IF(JJ.EQ.19)GO TO 680
    IF(ROUTE(II,JJ,KK).EQ.0)GO TO 680
    GO TO 660
650 II=ROUTE(II,20,KK)
    GO TO 645
690 IKEEP=II
    II=1
700 IF(ROUTE(II,20,KK).EQ.0)GO TO 690
710 II=II+1
    IF(II.GT.40)GO TO 1900
    GO TO 700
690 IF(II.EQ.IKEEP)GO TO 710
    JJ=1
720 IF(ROUTE(II,JJ,KK).EQ.0)JOINR)GO TO 730
    JJ=JJ+1
    IF(JJ.EQ.19)GO TO 710
    IF(ROUTE(II,JJ,KK).EQ.0)GO TO 710
    GO TO 720
730 JSAV=II
    ICAPCK=ROUTE(ISAV,21,KK)+ROUTE(JSAV,21,KK)
    IF(ICAPCK.GT.AMAX0(ROUTE(ISAV,22,KK),ROUTE(JSAV,
1 22,KK)))GO TO 740
    ROUTE(JSAV,22,KK)=AMAX0(ROUTE(ISAV,22,KK),ROUTE(JSAV,22,KK))
    ROUTE(ISAV,22,KK)=ROUTE(JSAV,22,KK)
-----

```

```

      JJ=JJ+1
      IF(JJ.GT.18)GO TO 736
      DO 735 JP=JJ,18
735  ROUTE(JS AV,JP, KK)=99999
736  ROUTE(JS AV,20, KK)=ISAV
      ROUTE(IS AV,19, KK)=JS AV
760  II=ROUTE(JS AV,19, KK)
      IF(II.EQ.0)GO TO 750
      JS AV=II
      GO TO 760
750  II=JS AV
751  ROUTE(II,21, KK)=ICAPCK
      IF(ROUTE(II,20, KK).EQ.0)GO TO 236
      II=ROUTE(II,20, KK)
      GO TO 751
0
C     THE POINT IS INFEASIBLE, ELIMINATE THE POINT
C
C     740 DIST(II,J,K)=0
C     GO TO 178
C
C-----
C
C     IF THE ROUTE MATRIX IS TOO SMALL FOR THE PROBLEM AN ERROR
C     MESSAGE IS WRITTEN. ENLARGING THE SIZE OF THIS MATRIX SHOULD
C     CORRECT IT. HOWEVER ALL THE PROPER STATEMENTS SHOULD BE
C     MODIFIED TO.
1900 WRITE(6,1701)I,J,K,II,JJ, KK,K3,MAXSAV, ICOST, ICAPCK
C
C     GO TO 2100
2000 CALL PEPEZ.RETURNS(128,1181)
C     SCAN TO MODIFY PANK IN SUCH A FORM THAT ONLY POINTS
C     THAT ARE ON THE ROUTE BE CONSIDERED FOR SUPPRESSION
C     THIS IS DUE TO THE METHOD USED WHEN RUNNING THE PROGRAM
C     WITH DIFFERENT TYPES OF TRUCKS
C-----
      IF(NUMTRK(2).EQ.0.OR.ICON .NE.0)GO TO 2010
      I=1
1803  II=RANK(I,1,K3)
      JJ=RANK(I,2,K3)
      IF(II.EQ.0)GO TO 2010
      IV=1
      JV=1
1805  IF(ROUTE(IV,JV, KK).EQ.0)GO TO 1830
      IF(ROUTE(IV,JV, KK).EQ.99999)GO TO 1810
      IF(ROUTE(IV,JV, KK).EQ.II)GO TO 1820
      JV=JV+1
      IF(JV.GT.18)GO TO 1810
      GO TO 1805
1830  IF(JV.EQ.1)GO TO 1831
1810  IV=IV+1
      JV=1
      GO TO 1805
1820  JV=JV+1
      IF(JV.GT.18)GO TO 1825
      IF(ROUTE(IV,JV, KK).EQ.0.OR.ROUTE(IV,JV, KK).EQ.99999)GO TO 1825
      IF(ROUTE(IV,JV, KK).EQ.JJ)GO TO 1860
1831  RANK(I,1,K3)=0
      RANK(I,2,K3)=0
      DO 1855 IP=I,99
      IF(RANK(IP+1,1,K3).EQ.0)GO TO 1808
      RANK(IP,1,K3)=RANK(IP+1,1,K3)
      RANK(IP,2,K3)=RANK(IP+1,2,K3)

```

```

1855 CONTINUE
1860 RANK(IP,1,K1)=0
      RANK(IP,2,K1)=0
      GO TO 1801
1860 I=I+1
      IF(I.GT.100)GO TO 1900
      GO TO 1803
1825 IF(ROUTE(IV,20,KK).EQ.0)GO TO 1831
      IF(ROUTE(ROUTE(IV,20,KK),1,KK).EQ.JJ)GO TO 1860
      GO TO 1831
C-----
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C           M E T H O D   O F   S U P P R E S S I O N S
C-----
2010 IF(ICONV.NE.0)GO TO 126
      IF(SUPP.EQ.0)GO TO 2100
      IF(K.NE.3)I4=I4+1
      IP4=I4-1
C
C   OBTAIN THE NEXT ELEMENT OF THE ORDERED JOINED PAIRS(RANK)
C
C   IF INITIAL SOLUTION RE-INITIALIZE VALUES AND RETURN
C
      N=RANK(I4,1,1)
      N1=RANK(I4,2,1)
      IF(K.NE.3)GO TO 2360
      IF(ITPOOL.NE.1)NEWCST=NEWCST-1000
      WRITE(6,132A)NEWCST
      DO 2354 I/P=1,SIZE
      DO 2354 J/P=1,SIZE
2354  DIST(I/P,J/P,3)=DIST(I/P,J/P,2)
      DIST(N,N1,3)=0
      IF(ISIMET.EQ.0)DIST(N1,N,3)=0
      DIST(N,N1,4)=0
      IF(ISIMET.EQ.0)DIST(N1,N,4)=0
      K=4
      KK=2
      K1=2
      ICOST=NEWCST
      GO TO 127
C
C   SEE IF THE NEW SOLUTION IS BETTER THAN THE CURRENT BEST
C   IN EITHER CASE SAVE THE BEST SOLUTION, RE-INITIALIZE THE
C   PROPER VALUES AND RETURN FOR THE NEXT ITERATION
C
2360 IF(ITPOOL.NE.1)NEWCST=NEWCST-1000
      IF(ICOST.GT.NEWCST)GO TO 2360
      WRITE(6,1329)IP4,NEWCST
377  CONTINUE
      IF(RANK(I4,1,1).EQ.0)GO TO 2100
      IF(I4.EQ.6)GO TO 2100
      DO 2370 M1=1,SIZE
      DO 2370 M2=1,SIZE
          DIST(M1,M2,4)=DIST(M1,M2,2)
2370  DIST(M1,M2,3)=DIST(M1,M2,2)
      DIST(N,N1,4)=0
      IF(ISIMET.EQ.0)DIST(N1,N,4)=0
      DIST(N,N1,3)=0
      IF(ISIMET.EQ.0)DIST(N1,N,3)=0
      DO 2375 I=1,40
      DO 2378 J=1,23
2378  ROUTE(I,J,2)=0

```

```

2380 GO TO 127
      CONTINUE
      WRITE(6,1330)I4
      WRITE(6,1331)NEWCST
      DO 2390 M1=1,SIZE
      DO 2390 M2=1,SIZE
2390   DIST(M1,M2,3)=DIST(M1,M2,4)
      ICOST=NEWCST
      DO 2400 M3=1,40
      DO 2400 M4=1,23
2400   ROUTE(M3,M4,1)=ROUTE(M3,M4,2)
      ROUTE(M3,M4,2)=0
      N=RANK(I4-1,1,1)
      N1=RANK(I4-1,2,1)
      DIST(N,N1,2)=0
      IF(ISIMET.F7.0)DIST(N1,N,2)=0
      I4=1
      DO 2410 M4=1,2
      DO 2410 M3=1,80
      RANK(M3,M4,1)=RANK(M3,M4,2)
2410   RANK(M3,M4,2)=0
      N=RANK(I4,1,1)
      N1=RANK(I4,2,1)
      GO TO 377

C
C*****
C      WRITE THE RESULTS FOUND AND TERMINATE IF APPROPRIATE
C*****
2100 WRITE(6,1332)ICOST
      WRITE(6,1333)
      DO 2500 I=1,40
2500   WRITE(6,1333)(ROUTE(I,J,1),J=1,22)
1304   FORMAT(I5)
      WRITE(6,1307)(NUMTRK(I),I=1,5)
      1307   FORMAT(5I6)
1301   FORMAT(10I10)
1303   FORMAT(14I4,4I15)
      1303A  FORMAT(25I5)
1310   FORMAT(F10,5)
1308   FORMAT(20I4/20I4/20I4/20I4/20I4/I4)
1302   FORMAT(20J4)
1306   FORMAT(6I5)
1319   FORMAT(14I,14X,"---OUTPUT FOR A NON-SYMMETRIC VEHICLE SCHEDULING P
%ROBLEM---")
1320   FORMAT(140,"THE DATA INPUT FOR THIS PROBLEM CONSISTS OF THE FOLLOW
%ING )
1321   FORMAT(" THE PROBLEM CONTAINS ".I3," DEMAND POINTS INCLUDING THE T
%RMINAL")
1322   FORMAT(" THE DEMAND AT EACH OF THE POINTS, N=2.....N, IS )
1323   FORMAT(// " THE SIZE AND NUMBER OF TRUCKS AVAILABLE ARE )
1324   FORMAT(" TRUCK SIZE ",7X,5I5)
1325   FORMAT(" NUMBER AVAILABLE ",5I5)
1327   FORMAT(14I,"FROM THE ABOVE COST MATRIX THE FOLLOWING SAVINGS MATRI
%X HAS BEEN CALCULATED )
1328   FORMAT(14I,"THE COST OF THE FIRST SOLUTION IS ",I7)
1329   FORMAT(140,"SUPPRESSION NUMBER ".I2," OF THIS SOLUTION GIVES A COS
%T OF ",I7)
1330   FORMAT(140,"SUPPRESSION NUMBER ".I2," HAS IMPROVED",
% THE SOLUTION")
1331   FORMAT(140,"THE NEW CURRENT BEST SOLUTION",
% HAS A COST OF ",I7)
1332   FORMAT(141,"THE FINAL SOLUTION HAS A COST OF ",I7,/// " THE ROUTES
%USED IN THIS SOLUTION ARE AS SHOWN BELOW")

```

```

1333  FORMAT(14H,"ROUTES",F8.6,"FROM",F10.1,"TO",
      X114,"DEMAND",F12.6,"CAPACITY")
1334  FORMAT(14H," THE TOTAL CPU TIME USED THUS FAR IS ",F9.5)
      JPROB=JPROB+1
      IF(JPROB.EQ.IPROB) GO TO 8618
      GO TO 8617
8618  CONTINUE
      EXTIME=SECOND(0.)-TIMEST
      WRITE(5,4099)EXTIME
4099  FORMAT(" THE TOTAL EXECUTION TIME IS", F15.4)
      STOP
      END

```

.....

SUBROUTINE IVAN, RETURNS(A1)

```

C
C   THIS SUBROUTINE READS IN THE DATA, AND MODIFIES IT IF
C   NECESSARY. IN THE CASE OF THE TRAVELING SALESMAN PROBLEM
C   IT ALSO SHIFTS ROWS AND COLUMNS WHEN PUNNING THE
C   TRAVELING SALESMAN PROBLEM IN SUCH A WAY THAT A SOLUTION
C   IS PRINTED FOR EACH NODE
C
      INTEGER C(85,85), SIZE, B
      INTEGER DEMAND(85), CAPCTY(5), SUPP, NUMTRK(5)
      * .RANK(100,2,2)
      * .DIST(85,85,4), ROUTE(40,23,2)
      LOGICAL SW, SWITCH
      COMMON //SIZE, JPROB, DEMAND, CAPCTY, SUPP, NUMTRK, IXRO
      * /BLOCK1/DIST, ROUTE, KK, K, L, C, SWITCH
      * /BLOCK3/ICON, SW, RANK, ICUAL, ICONT
      * /BLOCK5/ITRDEL
      SW=.FALSE.
      ICUAL=1
      ICONT=1
      K=3
      KK=1
      IF(JPROB.GT.0)GO TO 555
      SIZE IS THE NUMBER OF CITIES
      READ(5,*) SUPP, ICON
      READ(5,*) ITRDEL
C
C   SUPP INDICATES WHETHER THE SUPPRESSIONS ARE EXECUTED OR NOT
C   SUPP = 1   HAS SUPPRESSIONS
C   SUPP = 0   NO SUPPRESSIONS
C   ITRDEL IF ONE SPECIFIES A TRUCK DELIVERY PROBLEM
C   IF ZERO SPECIFIES A TRAVELING SALESMAN PROBLEM
      READ(5,*) (NUMTRK(I), I=1,5)
      IF(IXRO.NE.1)GO TO 555
      READ(5,*) (DEMAND(I), I=1,SIZE)
C
C   DEMAND IS THE DEMAND FOR EACH CITY
      READ(5,*) ((C(I,J), J=1,SIZE), I=1,SIZE)
C
C   C IS THE DISTANCE MATRIX
C   NUMTRK IS THE NUMBER OF TRUCKS FOR TYPE I TRUCKS
      READ(5,*) (CAPCTY(I), I=1,5)
C
C   CAPCTY IS THE CAPACITY FOR THE TRUCKS OF TYPE I
555  IF(.NOT.SWITCH.AND.IXRO.NE.1)RETURN
      DO 556 I=1,SIZE
      DO 556 J=1,SIZE
556  DIST(I,J,1)=C(I,J)
      IF(JPROB.EQ.0)GO TO 55
      IF(ITRDEL.EQ.1)RETURN 1
      DO 10 J=1,SIZE
      D=DIST(I,J,1)
      DIST(I,J,1)=DIST(JPROB+1,J,1)

```

```

10 DIST(I,JPROB+1,J,1)=B
   DO 40 I=1,SIZE
   B=DIST(I,1,1)
   DIST(I,1,1)=DIST(I,JPROB+1,1)
   DIST(I,JPROB+1,1)=B
40 CONTINUE
55 WRITE(6,111)
   DO 7 I=1,SIZE
   9 WRITE(6,111)(DIST(I,J,1),J=1,SIZE)
   IF(ITRDEL.EQ.1)RETURN
   DO 70 I=2,SIZE
70 DIST(I,1,1)=DIST(I,1,1)+500
   DO 80 J=2,SIZE
80 DIST(I,J,1)=DIST(I,J,1)+500
111 FORMAT(141,' THE FOLLOWING COST MATRIX IS BEING USED*')
11 FORMAT(25I5)
   RETURN
   END
   SUBROUTINE PEREZ, RETURNS(A1,A2)
C
C THIS SUBROUTINE TAKES CARE OF THE MULTIPLE TRUCK PROBLEM
C THE METHOD USED IS THE FOLLOWING
C INFINITE TRUCKS OF THE CAPACITY BEING USED ARE ASSUMED, AND AS
C MANY ROUTES AS NECESSARY ARE FORMED, THEN THE PROGRAM
C CALLS PEREZ WHICH FINDS THE SAVINGS THAT EACH ROUTE HAS
C AND KEEPS AS MANY ROUTES AS THERE ARE TRUCKS OF THAT TYPE
C PROVIDED THE FOLLOWING CONDITIONS ARE MET
C 1 - THE ROUTE NECESSARILY REQUIRES THIS TYPE OF TRUCK, I.E.
C IT CANNOT BE ASSIGNED TO A SMALLER TRUCK
C 2 - IT HAS SAVINGS LARGER THAN OTHER ROUTES, AND THUS SEEMS
C TO BE A BETTER CANDIDATE FOR THAT TRUCK
C
C THE SUBROUTINE DOES NOT FOOL AROUND WITH ONE TRUCK TYPE
C PROBLEM AND IF THIS IS THE CASE IT RETURNS WITHOUT
C MODIFYING ANYTHING.
COMMON //SIZE,JPROB,DEMAND,CAPCTY,SUPP,NUMTRK,IXRD/BLOCK1/
1 DIST,ROUTE,KK,K,L,C,SWITCH
* /BLOCK2//ICOST,JCOST,NEWGST
* /BLOCK3//ICOH,SW,PANK,ICUAL,ICONT
INTEGER DIST(95,4),ROUTE(40,23,2),DEMAND(85),NUMTRK(5)
1 .CAPCTY(5),COST(40),ROW(40),SIZE
2 .PANK(100,2,2)
3 .C(85,85)
LOGICAL SW,SWITCH
DATA ICOST(I),I=1,40//40*0/,MARK/1/
IF(L.EQ.5.OR.NUMTRK(L+1).EQ.0)GO TO 1500
DO 1501 I=1,40
1501 COST(I)=0
IR=1
IF=MARK
1 IF(ROUTE(II,19,KK).NE.0)GO TO 2
IF(ROUTE(II,1,KK).EQ.0)GO TO 3
ROW(IP)=II
IP=IR+1
2 II=II+1
GO TO 1
3 IR=IR-1
IF(IR.LE.NUMTRK(L))GO TO 1500
DO 5 INUM=1,IR
II=ROW(INUM)
9 DO 7 J=1,17
IF(ROUTE(II,J+1,KK).EQ.3.OR.ROUTE(II,J+1,KK).EQ.99999)GO TO 8
COST(INUM)=COST(INUM)+DIST(ROUTE(II,J,KK),ROUTE(II,J+1,KK),2)
7 CONTINUE

```

```

8 IF (ROUTE(II,20,KK).EQ.0) GO TO 6
COST(IP)=COST(IP)+DIST(ROUTE(II,J,KK),ROUTE(ROUTE(II,20,KK),I,KK),
12)
II=ROUTE(II,20,KK)
GO TO 9
6 CONTINUE
INUMTRK=NUMTRK(L)
DO 11 IPX=1,INUMTRK
MAX=0
ILINE=0
DO 12 IPXX=1,IR
IF(ROUTE(ROW(IPXX),21,KK).LE.CAPCTY(L+1)) GO TO 12
IF(COST(IPXX).LE.MAX) GO TO 12
MAX=COST(IPXX)
ILINE=IPXX
12 CONTINUE
IF(IILINE.EQ.0) GO TO 114
COST(IILINE)=-99999
11 CONTINUE
114 DO 13 IPX=1,IR
IF(COST(IPX).EQ.-99999) GO TO 13
II=ROW(IPX)
20 DO 14 J=1,19
14 ROUTE(II,J,KK)=0
IF(ROUTE(II,20,KK).EQ.0) GO TO 15
II=ROUTE(II,20,KK)
DO 15 J=19,23
16 ROUTE(ROUTE(II,19,KK),J,KK)=0
ROUTE(ROUTE(II,19,KK),1,KK)=-1
GO TO 20
15 DO 17 J=19,23
17 ROUTE(II,J,KK)=0
ROUTE(II,1,KK)=-1
13 CONTINUE
IC=1
DO 30 I=MARK,40
IF(ROUTE(I,1,KK).EQ.0) GO TO 35
IF(ROUTE(I,1,KK).NE.-1) GO TO 30
DO 41 IPX=1,40
145 IF(ROUTE(IPX+IC,1,KK).EQ.0) GO TO 35
IF(ROUTE(IPX+IC,1,KK).EQ.-1) GO TO 140
GO TO 46
140 IC=IC+1
GO TO 145
46 DO 43 J=1,22
43 ROUTE(IPX,J,KK)=ROUTE(IPX+IC,J,KK)
IF(ROUTE(IPX,19,KK).EQ.0.AND.ROUTE(IPX,20,KK).EQ.0) GO TO 40
DO 43 IV=1,40
IF(ROUTE(IV,1,KK).EQ.0) GO TO 40
IF(ROUTE(IV,19,KK).EQ.IPX+IC) ROUTE(IV,19,KK)=IPX
IF(ROUTE(IV,20,KK).EQ.IPX+IC) ROUTE(IV,20,KK)=IPX
49 CONTINUE
40 CONTINUE
30 CONTINUE
35 MARK=IPX
INO=IPX+IC
DO 1700 I=MARK,INO
DO 1700 J=1,23
1700 ROUTE(I,J,KK)=0
IU=2
IF(K.F0.4) IU=3
IF(ICON.NE.0) IU=2
NENCGST=JCOST
DO 50 II=1,SIZE

```

```

DO 50 JJ=1,SIZE
50 DIST(II,JJ,K)=DIST(II,JJ,IU)
IKEEP=0
II=1
53 IF(ROUTE(II,19,KK).NE.0)GO TO 62
IF(ROUTE(II,1,KK).EQ.0)GO TO 1000
IF(II.EQ.IKEEP)GO TO 60
IKEEP=II
56 DO 65 J=1,17
I=ROUTE(II,J,KK)
IJ=ROUTE(II,J+1,KK)
IF(IJ.EQ.0.OR.IJ.FQ.99999)GO TO 66
DIST(I,1,K)=0
DIST(I,IJ,K)=0
DIST(I,IJ,K)=-1
DIST(IJ,I,K)=0
NEWOST=NEWOST-DIST(I,IJ,2)
65 CONTINUE
J=18
66 IF(ROUTE(II,20,KK).EQ.0)GO TO 60
I=ROUTE(II,J,KK)
II=ROUTE(II,20,KK)
IJ=ROUTE(II,1,KK)
DIST(II,1,K)=0
DIST(II,IJ,K)=0
DIST(II,IJ,K)=-1
DIST(IJ,I,K)=0
NEWOST=NEWOST-DIST(II,IJ,2)
GO TO 56
60 II=IKEEP+1
GO TO 53
62 II=II+1
GO TO 53
1000 L=L+1
IF(L.GT.5)GO TO 1500
IF(NUMPK(L).FQ.0)GO TO 1500
DO 1600 I=MARK,40
1600 ROUTE(I,22,KK)=CAPCTY(L)
IF(ICDN.NE.0.AND.DIST(RANK(ICUAL-1,1,1),RANK(ICUAL-1,2,1),K).NE.-1)
1 GO TO 1601
RETURN1
1601 JOINR=RANK(ICUAL-1,1,1)
JOINC=RANK(ICUAL-1,2,1)
RETURN2
1500 L=1
MARK=1
RETURN
END
SUBROUTINE MARIA,PETURNS(A1,A2)

```

```

C
C   THIS SUBROUTINE IS USED TO EXECUTE THE METHOD OF
C   M A X I M U M   S A V I N G S
C   - - - - -
C   THE METHOD PROCEEDS AS FOLLOWS:
C   UPON THE FIRST CALL THE POINT OF EACH COLUMN AND ROW, STARTING
C   WITH COLUMN 2 ROW 2, WHICH PRESENTS THE MAXIMUM SAVINGS
C   IS SAVED IN THE MATRIX RANK, PROVIDED SUCH A POINT IS
C   NOT ALREADY IN THE MATRIX, OR IF SO CHOSEN, ITS OPPOSITE
C   ALSO.
C   THE POINTS IN THE MATRIX ARE THEN ORDERED SO THAT THE
C   ONE HAVING THE HIGHEST SAVINGS GOES FIRST, THE ONE HAVING THE
C   SECOND HIGHEST SAVINGS GOES SECOND, AND SO ON
C   IN THIS MANNER AS MANY POINTS AS DESIRED ARE ORDERED.

```

```

C   WHEN THE SUBROUTINE IS SUBSEQUENTLY CALLED THE POINT
C   COMING NEXT IN THE MATRIX RANK IS GIVEN BACK TO THE
C   MAIN PROGRAM TO BE USED AS THE FIRST POINT ASSIGNED ON
C   ROUTE. HOWEVER THE SAVINGS MATRIX REMAINS UNTOUCHED.
COMMON //SIZE, JPRD, DEMAND, CAPCTY, SUPP, NUMTRK, IKRO
* /BLOCK1/DIST,ROUTE,KK,<.L.C.SWITCH
* /BLOCK2/ICOST,JCOST,NCWCST
* /BLOCK3/ ICON,SW,RANK,ICUAL,ICONT
* /BLOCK4/JOINR,JOINC
* /BLOCK5/ITRDEL
* /BLOCK6/ISIMFT
INTEGER SIZE,DEMAND(85),CAPCTY(5),SUPP
* ,NUMTRK(5),RANK(100,2,2)
* ,DIST(85,85,4),ROUTE(40,23,2)
* ,C(85,85)
LOGICAL SW,SWITCH
IF(ICON.F0.0)RETURN
IF(SW)GO TO 7
I=2
11 MAX=DIST(I,I)
DO 2 J=1,SIZE
IF(DIST(I,J,2).LE.MAX)GO TO 2
MAX=DIST(I,J,2)
IROW=I
ICOL=J
2 CONTINUE
J=I
DO 1 II=2,SIZE
IF(DIST(II,J,2).LE.MAX)GO TO 1
MAX=DIST(II,J,2)
IROW=II
ICOL=J
1 CONTINUE
IF(MAX.EQ.0)GO TO 4
IF(DEMAND(IROW)*DEMAND(ICOL).GT.CAPCTY(1))GO TO 3
IF(DIST(IROW,ICOL,1).EQ.-1)GO TO 4
DIST(IROW,ICOL,1)=-1
IF(ISIMFT.EQ.0)DIST(ICOL,IROW,1)=-1
15 RANK(ICONT,1,1)=IROW
RANK(ICONT,2,1)=ICOL
ICONT=ICONT+1
GO TO 4
3 DIST(IROW,ICOL,2)=0
GO TO 4
5 ICONT=ICONT-1
INUM=ICONT-1
INUM=AMIND(INUM,ICONT)
DO 50 I=1,INUM
MAX=DIST(RANK(I,1,1),RANK(I,2,1),2)
II=I+1
LOC=I
DO 60 IF=II,ICONT
IF(DIST(RANK(IF,1,1),RANK(IF,2,1),2).LE.MAX)GO TO 60
LOC=IF
MAX=DIST(RANK(IF,1,1),RANK(IF,2,1),2)
60 CONTINUE
IP1=RANK(I,1,1)
IP2=RANK(I,2,1)
RANK(II,1,1)=RANK(LOC,1,1)
RANK(II,2,1)=RANK(LOC,2,1)
RANK(LOC,1,1)=IP1
RANK(LOC,2,1)=IP2
50 CONTINUE
SW=.TRUE.

```

```

7 IF(ICUAL.GT.ICONT.OR.ICUAL.GT.ICON)GO TO 102
  JOIN2=RANK(ICUAL,1,1)
  JOIN3=RANK(ICUAL,7,1)
  IF(ICUAL.EQ.1)GO TO 75
  IF(ICUAL.EQ.2)GO TO 76
  IF(ITRDEL.NE.1)NEWCST=NEWCST-1000
  IF(NEWCST.LT.ICOST)GO TO 90
  WRITE(6,79)ICUAL-1,NEWCST
79 FORMAT(" THE MAXIMUM SAVINGS",I3," HAS A COST OF",I10)
92 DO 90 M1=1,SIZE
  DO 90 M2=1,SIZE
90 DIST(M1,M2,K1)=DIST(M1,M2,2)
  DO 85 M1=1,40
  DO 85 M2=1,22
85 ROUTE(M1,M2,KK)=0
  NEWCST=JCOST
  ICUAL=ICUAL+1
  RETURN1
75 K=3
  KK=1
  L=1
  ICUAL=ICUAL+1
  NEWCST=JCOST
  RETURN1
76 K=4
  IF(ITRDEL.NE.1)NEWCST=NEWCST-1000
  WRITE(6,79)ICUAL-1,NEWCST
  KK=2
  L=1
  ICOST=NEWCST
  ICUAL=ICUAL+1
  NEWCST=JCOST
  RETURN1
90 WRITE(6,190)ICUAL-1,NEWCST
190 FORMAT(" POINT",I3," IMPROVED THE SOLUTION,THE NEW COST IS",I10)
  ICOST=NEWCST
  IF(K.EQ.4)GO TO 101
  K=4
  KK=2
  L=1
  GO TO 92
101 K=3
  KK=1
  L=1
  GO TO 92
102 IF(ITRDEL.NE.1)NEWCST=NEWCST-1000
  IF(NEWCST.LT.ICOST)GO TO 105
  IF(K.EQ.4)GO TO 110
  KK=2
115 DO 103 II=1,40
  DO 103 JJ=1,22
103 ROUTE(II,JJ,1)=ROUTE(II,JJ,2)
110 WRITE(6,79)ICUAL-1,NEWCST
  RETURN2
105 ICOST=NEWCST
  IF(K.EQ.4)GO TO 115
  GO TO 110
4 I=I+1
  IF(I.GT.SIZE)GO TO 5
  GO TO 11
END
SUBROUTINE RAND(ISEED,LIMIT,X)
C
C   THIS SUBROUTINE GENERATES A SIZE 'SIZE' VECTOR OF UNIFORMLY
C
C   DISTRIBUTED RANDOM NUMBERS, GIVEN A STARTING SEED.
C
  DIMENSION X(101)
  DO 1 I=1,LIMIT
  ISEED=ISEED*16777219
  IF(ISEED.LT.0)ISEED=-ISEED
  X(I)=ISEED/(2.**44.)
1 CONTINUE
  RETURN
  END

```

REFERENCES

1. Ashour, Vega and Parker, "A Heuristic Algorithm for Traveling Salesman Problem," Transpn. Res., Vol. 6, pp. 187-195, 1971.
2. Baker, K. R., "Introduction to Sequencing and Scheduling," Duke University, 1974.
3. Balinski, M. L. and R. E. Quant, "On an Integer Program for 2 Delivery Problem," Operation Research, Vol. 12, No. 2, 1964.
4. Bellman, R., "Dynamic Programing Treatment of the Traveling Salesman Problem," J. Assoc. Comp. Mech., 9, 61-63 (1962).
5. Bellmore, M., and Malone, J. C., "Pathology of Traveling Salesman Subtour Elimination Procedure," Operation Research, 19, 2 (March-April 1971).
6. Clarke, G. and W. Wright, "Scheduling of Vehicles from a Central Depot to a Number of Delivery Points," Operation Research, Vol. 12, 1964.
7. Cochran, H., "Optimization of a Carrier Routing Problem," M. S. Thesis, Kansas State University, Manhattan, Kansas.
8. Conway, R. M., Maxwell, W. L. and Miller, L. W., "Theory of Scheduling," Addison-Wesley, Reading Massachusetts.
9. Christofides, N. and S. Eilon, "An Algorithm for the Vehicle Dispatching Problem" Operational Research Quarterly, Vol. 20, 1969.
10. Dantzing, G. B., and Rawser, J. H., "The Truck Dispatching Problem," Management Science, Vol. 6, No. 1, 1959.
11. Dantzing, G. B., D. R. Fulkerson, and S. M. Johnson, "Solution of a Large-Scale Traveling Salesmen Problem," Operation Research, Vol. 2, 1954.
12. Eastman, W. L., "A Solution to the Traveling Salesman Problem," Presented at the American Summer Meeting of the Econometric Society, Cambridge, Massachusetts (August 1958).
13. Ford, L. R. and D. R. Fulkerson, Flows in Networks, Princeton University Press, 1962.
14. Garving, W. W., H. W. Crandell, J. B. John, and R. A. Spellman, "Application of Linear Programming in the Oil Industry," Management

Science, Vol. 3, 1957.

15. Gaskell, T. J., "Bases for Vehicle Fleet Scheduling," Operational Research Quarterly, Vol. 18, No. 3, 1967.
16. Gillete, B. E. and L. R. Miller, "A Heuristic Algorithm for the Vehicle-Dispatch Problem," Operation Research, Vol. 22, No. 2, 1974.
17. Gomory, R. E., "Outline of an Algorithm for Integer Solutions of Linear Problems," Bulletin of the American Mathematical Society, 64, 1958.
18. Hayes, R. L., "The Delivery Problem," Master's Thesis, Carnegie Institute of Technology, 1967.
19. Held, M. and R. M. Karp, "The Traveling Salesman Problem and Minimum Spanning Trees," Operation Research, Vol. 18, No. 6, 1970.
20. Held, M. and R. M. Karp, "A Dynamic Programming Approach to Sequencing Problems," SIAM 10, 196-210.
21. Holmes, R. A., "The Routing and Scheduling of a Class of Postal Vehicles," Master's Thesis, Georgia Institute of Technology, Atlanta, Georgia, 1975.
22. Karg, R. L. and G. L. Thompson, "A Heuristic Approach to Solving Traveling Salesman Problems," Management Science, No. 10, 1964.
23. Lin, S., "Computer Solutions of the Traveling Salesman Problem," Bell System Technical Journal, Vol. 44, 1965.
24. Lin, S. and B. W. Kernighan, "An Effective Heuristic Algorithm for the Traveling Salesman Problem" Operation Research, Vol. 21, 1973.
25. Little, J. D. C., K. G. Murty, D. W. Sweeney, and C. Karel, "An Algorithm for the Traveling Salesmen Problem," Operation Research, Vol. 4, No. 6, 1963.
26. Miller, C., Tucker, A., and Zemlin, R., "Integer Programming Formulation and Traveling Salesman Problem," T. Ass. Comput. Mech., 7, 326-329 (1960).
27. Montgomery, D. C. and Johnson, L., "Operation Research in Production Planning, Scheduling and Inventory Control," Georgia Tech. 1974.
28. Newton, K. and W. Thomas, "Developing a Computer Program for Bus Routing: Final Report," State University of New York, Research Foundation, Albany, New York, July 1970.

29. Pierce, J. F., "Direct Search Algorithms for Truck-Dispatching Problems," Transportation Research, Vol. 3, No. 1, 1969.
30. Raymond, T. C., "Heuristic Algorithm for the Traveling Salesman Problem," IBM J. Res. Devel., 400-407, 1969.
31. Svestka, J. and V. Huckfeldt, "Computational Experience with an M-Salesman Traveling Salesman Algorithm," Management Science, Vol. 19, 1973.
32. Tillman, F. A. and H. Cochran, "A Heuristic Approach for Solving the Delivery Problem," Journal of Industrial Engineering, Vol. 19, No. 7, 1968.
33. Wagner, H. M., "Principles of Operation Research," Prentice-Hall, Englewood Cliffs, New Jersey, (1969).
34. Wayne, T., M. Ghare, L. Fourdi, "Transportation Routing Problem - A Survey," AIIE Transactions, Vol. 6, No. 4, 1974.