THIS BOOK IS OF POOR LEGIBILITY DUE TO LIGHT PRINTING THROUGH OUT IT'S ENTIRETY.

THIS IS AS RECEIVED FROM THE CUSTOMER.

A HEURISTIC ALGORITHM FOR TRAVELING SALESMAN PROBLEMS

by

JUAN FELICIANO VEGA

B.S., National University of Engineering, Peru, 1961

A MASTER'S THESIS

submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

Department of Industrial Engineering

KANSAS STATE UNIVERSITY

Manhattan, Kansas

1970

Said Ashour

Major Professor

THIS BOOK CONTAINS NUMEROUS PAGES WITH THE ORIGINAL PRINTING BEING SKEWED DIFFERENTLY FROM THE TOP OF THE PAGE TO THE BOTTOM.

THIS IS AS RECEIVED FROM THE CUSTOMER.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS

The author is deeply indebted to his major advisor, Dr. Said Ashour, for his invaluable guidance and assistance which made possible this thesis.

The author also wishes to express his appreciation to Mrs. Marie Jirak for her assistance and excellent work of typing this thesis.

CHAPTER I

INTRODUCTION

Combinatorial problems are defined as those which consist of determining a solution among a set of alternatives such that a specific criterion is optimized subject to a given set of constraints. The number of feasible solutions from which the optimal has to be selected constitutes a finite set. Complete enumeration of all feasible solutions of a combinatorial problem is theoretically feasible. In practice, however, it can be applied only to problems of small size because of the computational effort involved and the storage required.

The objective of finding efficient methods of solution for combinatorial problems has led to the development of various approaches. One characteristic common to all approaches is the provision of means to reduce the number of alternatives to be investigated, thus reducing the computation time.

The variables involved in combinatorial problems are integer valued, of the zero-one type. The number of variables, and consequently, the number of feasible solutions, increases much faster than the size of the problem under consideration.

Delivery, or carrier routing, problems consist in the determination of assignments of trucks, or carriers, into routes originated at given depot locations in order to satisfy the demands of a number of customers placed at known locations, provided that capacity, tour length, or any other constraints imposed on the system are not violated, such that the total distance (or cost) incurred is a minimum. This definition is very general and encompasses a series of different types of problems, as

illustrated in Table 1.1. It should be pointed out that the name delivery problem has been generally accepted as a substitute for the single terminal delivery problem; this convention will be respected in this thesis.

1.1 Delivery problem

The delivery problem consists of finding a set of routes, all starting and ending at the same terminal, in order to satisfy the demands of a given set of customers such that the constraints imposed on the system are not violated and the objective function, such as the total distance traveled, is minimized. In order to develop the mathematical model for the delivery problem, the following assumptions [22] are made:

1. The distance matrix is symmetrical.
2. Demands at each point are deterministic and of known value.
3. Each demand point, or city, is included in only one route.
4. Capacity of carriers is known.

The problem may be interpreted as one of the delivering or collecting commodities, exclusively. For mathematical presentation, consider the following notation:

n,	number of demand points, or cities, not including the depot which is designated as point 0,
$C(i,j)$,	distance for traveling from point i to point j,
H,	number of trucks assigned on routes,
Q_h,	freight capacity of carrier h,
q_i,	demand at point i,
D,	maximum route length, for all carriers,
$A(i,j)$,	assignment or decision variable such that

TABLE 1.1 Types and General Characteristics of Delivery Problems

Problem Denomination	Number Terminals		Function of Terminals		Capacity of Carrier		Number of Routes in Solution	
	Single	Multiple	Collect or Delivery	Collect and Delivery	Greater or Equal to Total Load	Less Than Total Load	One	One or More Than One
Traveling salesman problem	X		X		X		X	
Single trip delivery problem	X		X		X		X	
Heterogeneous fleet delivery problem	X		X			X		X
Common carrier delivery problem	X		X			X		X
Multiple terminal delivery problem		X	X			X		X
Live haul delivery problem		X		X		X		X

$$A(i,j) = \begin{cases} 1, \text{ if demand points } i \text{ and } j \text{ are paired on} \\ \text{the same route} \\ 0, \text{ otherwise.} \end{cases}$$

Thus, the delivery problem can be formulated mathematically as follows:

Minimize

$$\sum_{i \neq j} C(i,j)\, A(i,j), \qquad i, j = 0, 1, 2, \ldots, n, \tag{1}$$

subject to:

$$\sum_{j=0}^{k-1} A(k,j) + \sum_{i=k+1}^{n} A(i,k) = 2; \qquad k = 1, 2, \ldots, n \tag{2}$$

$$\sum^{h} q_i \leq Q_n; \qquad h = 1, 2, \ldots, H \tag{3}$$

$$\sum^{h} [C(i,j)A(i,j)] \leq D; \qquad h = 1, 2, \ldots, H \tag{4}$$

$$A(i,j)_{i \neq j} = 1 \text{ or } 0, \qquad i, j = 0, 1, 2, \ldots, n \tag{5}$$

The constraint (2) establishes that each point is included in only one route. The constraint (3) imposes the capacity limitations for every truck h. The maximum distance traveled by every truck is established by constraint (4). This model, though simple, gives rise to a great number of constraints and variables, and hence, to the difficulties encountered in solving the problem. For instance, if n = 10, and H = 3, the number of constraints imposed on the system can be determined as follows:

1. From constraint (2), for each value of k an inequality is constructed, that is, constraint (2) gives rise to n, or 10, specific constraints.

2. The combined load of all the points assigned, to the same route should not exceed the capacity of the carrier, thus, H, or 3, constraints result from constraint (3),
3. The total length of each route should not exceed the established limit, D; as a consequence, one constraint is constructed for every route, and H, or 3, constraints result from (4),
4. The number, N, of zero-one constrained assignment variables is equal to the number of cells below the main diagonal of a (n+1) matrix, thus for n = 10,

$$N = \frac{(n+1)^2 - (n+1)}{2}$$

$$= \frac{11^2 - 11}{2}$$

$$= 55$$

Thus, for the single problem referred to here, the total number of constraints is 71. This high number of constraints in comparison with the size of the problem is one of the reasons that allows us to state why the solution to the problem is so difficult when finding of the optimal solution is attempted.

Methods of solution. Several approaches have been devised for solving the preceding model. These methods can be classified into two general types, that is, optimal- and suboptimal-searching methods.

Optimal searching procedures are those which after a finite number of steps will discover an optimal solution(s) for the given problem. The advantage of achieving optimality is frequently overcome by the great amount of computational time involved. Three optimal searching procedures

have been applied to the delivery problem with relatively small success, namely, dynamic programming, integer linear programming and combinatorial programming.

1. Dynamic programming. This is a stage-by-stage computational procedure applicable to constrained optimization problems in which each stage is identified by a state variable. The decision variables and their respective constraints are grouped according to their corresponding stages and these are considered sequentially. The main characteristic of dynamic programming algorithms is the recursive nature of the computational procedure. Several investigators [4, 21] have applied dynamic programming algorithms to delivery problems, solving problems with up to thirteen cities. Unfortunately, the computer time required for solution grows much faster than exponentially with the number of cities involved [19], and, simultaneously, storage requirements become excessive when using the computer for larger problems. Little [25] has noted that the solution by dynamic programming of a 20-city delivery problem will take about 10 hours on an IBM 7090 computer; even though, storage requirements will prohibit the attempt of solution.

2. Integer linear programming. The nature of the objective function and constraints imposed on the delivery problem makes it suitable for solution by integer linear programming. However, the results obtained in terms of computer time required are not generally encouraging. The problem has to be formulated first as an assignment problem, and appropriate restrictions are then imposed. The best known formulation of the delivery problem can be taken as a special case of the formulation given by Elson

and Hellerman [15] for the solution of the line haul problem. For certain problems, it is possible to find an integer-valued solution without any integer restriction [18], as for the assignment, transportation, and network flow problems [5]. The requirement is that the demands be integer-valued.

3. Combinatorial programming. This approach is based on two fundamental concepts, namely, the controlled enumeration of potential solutions, which can be done, at least, implicitly and avoid explicit consideration of those solutions which fail to be satisfactory due to considerations such as bounding, dominance, and feasibility. Using these concepts throughout the problem solving procedure, the achievement of an optimal solution(s) is guaranteed. The use of combinatorial programming [5] has three desirable characteristics:

1. It is possible to obtain usable solutions before conclusion of the problem solving procedure.
2. The region to be searched can be reduced by the use of available bounds.
3. A number of solutions can be generated including the possibility of finding those deviating a certain amount from the optimal solution.

Little et. al. [26] have developed a technique known as branch-and-bound for the solution of the traveling salesman problem. Hayes [20] devised a branch-and-bound algorithm for solving the delivery problem reporting little success in doing so. The approach of Hayes is to transform the delivery problem into a traveling salesman problem, and then apply an algorithm similar to that of Little et. al., taking care of the appropriate constraints proper of the delivery problem.

Suboptimal searching procedures are those developed on the base of heuristic rules in order to reduce the amount of computational effort with the object of finding an optimal or near-optimal solution to the given problem. A heuristic procedure does not respond in the same way for all problems, depending upon the structure of the problem. Heuristic procedures for the solution of the delivery problem belong to one of the following four main categories:

1. Heuristic methods based on savings of distances.
2. n-optimal heuristic methods.
3. Heuristic methods based on decisions taken under consideration of several weighted factors.
4. Other types of heuristic methods.

1. Heuristic methods based on savings. Dantzig and Ramser [13] have implemented the first known heuristic algorithm for the solution of the delivery problem, under the name of truck dispatching problem. Their approach is based on some stages of aggregation in which a number of customers are grouped such that their joint demand does not exceed the capacity allowed at the given stage of aggregation. Improvements in the construction of routes are made by eliminating certain combinations which give a larger number of carriers or longer route lengths. The method as applied to their sample problem results in a near optimal solution.

Clarke and Wright [8] have used an approach similar to the above but eliminated the restrictions on the stages of aggregation. They have developed a heuristic algorithm in which the selection criterion for inclusion of a pair of delivery points on the same route is the highest

savings in distance traveled. At the outset, a carrier is assigned to every demand point, constituting what could be thought of as an upper bound on the total distance traveled. Then, the routes are formed stage by stage through maximizing the savings gained by joining pairs of cities in the routes, such that the capacity constraints are not violated. The distance saving achieved by linking points I and J, with respect to the terminal, 0, is evaluated such that

$$S(I,J) = C(0,K) + C(J,0) - C(I,J).$$

Cochran [9] has formulated a heuristic method based on the procedure of Clarke and Wright, which allows more flexibility in the handling of additional constraints. Thus, the route length constraints can be added. The assignment of demand points is made such that the combined loads will be assigned to the smallest truck capable of handling this demand.

Tillman and Cochrane [34] applied a look ahead rule for the assignment of priorities in order to establish which pair of points will be included in the route under consideration. The procedure is basically the same as that of Clarke and Wright with the extension suggested by Cochran. The feature of the look ahead rule has resulted in a better solution.

Tillman and Hwang [35] have devised an extension of the preceding method for the solution of the multiple terminal delivery problem. The savings were corrected according to the closest terminal for each city. This correction was necessary because the savings, as computed previously, are larger when each of the linked points is farther from the terminal.

2. n-optimal decision methods. These methods involve the generation of routes and the exchange of links leading to and leaving from n points

by a new set of links such that the new total distance is smaller than the original one. The procedure [7] is repeated many times and at each iteration the best solution is saved and used as the starting route for the next iteration. The probabilities of obtaining the optimal solution are improved every time. Most algorithms using this approach have been devised for the traveling salesman problem and will be discussed later in the next section.

3. Weighted scores method. Based on Karg and Thomson's approach, Hayes [20] has developed a heuristic method to decompose the delivery problem into two subproblems: (1) determine which customers are to be assigned on each route, and (2) apply a traveling salesman algorithm to each route so as to minimize the total distance traveled. An outside point is defined as a customer, or demand point, located at the periphery of the area served by the terminal. The customer farthest from the terminal is selected as the first outside point. The second outside point is chosen such that the product of its distance from the terminal and its distance from the first outside point is maximum. The remaining outside points are selected such that the product of its distance from the terminal and its distance to all other demand points is maximum. These outside points together with the terminal are then used to construct the routes. Demand points are associated with outside points in respective routes upon consideration of scores computed on the basis of demand, number of unplaced customers within a specified distance, distance from a straight line between the initial point and the terminal, distance from the terminal, and distance from the nearest other outside point. The assignment of customers in the respectives routes is made according to the maximum score.

4. Other types of heuristic methods. Using a trial and error approach, Gaskel [17] has found the solution for several delivery problems and found the optimal routes for some of them. However, no rules can be given for his trial and error approach. It is rather interesting to note that for some problems, it is quite possible to find a near-optimal or optimal solution by observing the graphical representation of the terminal and demand points. Gaskell's approach cannot be applied when the configuration of the problem is not Euclidean, that is, when the distances among the three points I, J, and K do not satisfy the triangle inequality

$$\overline{IK} \leq \overline{IJ} + \overline{JK}$$

for any I, J, and K, where all distances are considered to be positive.

1.2 Traveling Salesman Problem

The traveling salesman problem consists of finding the minimal distance route starting and ending at the same terminal, and visiting, once and only once, each of a number of cities. There are several problems having a similarity in the mathematical model with that of the traveling salesman problem. For example, problems which are related with finding the optimal sequence of jobs on a single machine, where the cost or time employed is affected by the precedence order of the jobs. Routing of school buses can be treated with the same technique used for solving the traveling salesman problem.

The mathematical formulation of the traveling salesman problem involves the following assumptions:

1. The distance (or cost) matrix can be symmetrical or non-symmetrical.

2. Demands at each city (or demand point) are assumed having a value of unity.

3. The capacity of the salesman (or carrier) is large enough to attend all cities (or demand points) in one single trip.

As discussed above, the problem may be interpreted as one of delivering or collecting commodities, exclusively. Let us now consider n as the number of cities to be visited, including the depot. Using the notation previously given in Section 1.1, the objective function is such that:

Minimize

$$\sum_{i=1}^{n} \sum_{j=1}^{n} C(i,j)\, A(i,j), \qquad i \neq j, \tag{6}$$

subject to:

$$\sum_{j=1}^{n} A(i,j) = 1, \qquad i = 1, 2, \ldots, n, \tag{7}$$

$$\sum_{i=1}^{n} A(i,j) = 1, \qquad j = 1, 2, \ldots, n, \tag{8}$$

$$A(i,j) = 1 \text{ or } 0.$$

Constraints resulting from relations (7) and (8) state that the salesman arrives to and departs, once and only once, from each city. The assignment variable A(i,j) establishes that city i and city j are paired if A(i,j) = 1; if they are not linked directly, then A(i,j) = 0. It is implied that the solution should be a complete tour, that is, the salesman must visit all the cities before returning home. Such a restriction can be expressed mathematically in several different ways, as it will be seen later on in this Chapter.

Methods of Solution. Bellmore and Neuhauser [5] have considered the solution of the traveling salesman problem to be composed of three parts: (1) a starting point, (2) a solution generation scheme, and (3) a termination rule. The method of solution is considered exact when the termination rule is such that the iteration stops if and only if the solution is optimal. If the termination rule is such that the iteration stops not only if the solution is optimal, the solution is approximate. Based on these ideas, the review of the literature is centered mainly on the solution generation scheme. This is, because starting points and termination rules depend on that scheme. Solution generating procedures are of three principal types, namely: tour-to-tour improvement, tour building and subtour elimination.

Tour-to-tour improvement procedures start with an arbitrary tour. The solution generation scheme consists of identifying another tour which is slightly different from the actual tour, such that, an improved value of the objective function is obtained. This procedure is terminated when new tours are found, such that no improvement of the objective function results. Tour-to-tour improvement solution generation schemes have been applied to the development of heuristic algorithms. To the author's knowledge, no exact method using this type of approach is known.

Tour building procedures start identifying one initial node. Links are then assigned successively in such a way that the complete tour is constructed. This type of solution generating scheme has been applied to the development of both, exact and approximate, methods of solution.

Subtour elimination procedures start with the optimal solution to the assignment problem corresponding to the distance matrix of the given

traveling salesman problem. Whenever the optimal solution to the assignment problems constitutes a complete tour, an optimal solution has been found for the traveling salesman problem. If it includes subtours, these have to be eliminated by some specific set of rules. There is a number of exact solution methods using the subtour elimination approach.

Exact methods of solution for the traveling salesman problem fall into one of the three types of techniques: (1) dynamic programming, (2) integer programming, and (3) branch-and-bound.

Dynamic programming. A stage-by-stage tour building recursive procedure is applied such that if $f_{k-1}(P_{j_m} | P_{j_1}, P_{j_{m-1}}, P_{j_{m+1}}, \ldots, P_{j_{k-1}})$ is the shortest path from city P_1 to city P_{j_m} passing through cities $(P_{j_1}, \ldots, P_{j_{m-1}}, P_{j_{m+1}}, \ldots, P_{j_{k-1}})$, the shortest partial tour from city P_{j_1} to city P_j can be determined such that

$$f_k(P_j | P_{j_1}, \ldots, P_{j_{k-1}}) =$$

$$\min_{m=1,\ldots,k-1} [f_{k-1}(P_{j_m} | P_{j_1}, \ldots, P_{j_{m-1}}, P_{j_{k-1}}) + C(P_{j_m}, P_j)]$$

that is, the shortest path from city P_{j_1} to city P_j is equal to the smallest of the preceding routes plus their respective distance $C(P_{j_m}, P_j)$, for $m = 1, \ldots, k-1$. Thus, at the start, partial routes are evaluated for each pair of cities according to the relations:

$$f_2(P_{j_k} P_{j_1}) = C(1, P_{j_1}) + C(P_{j_1}, P_{j_k}),$$

$$\text{for all } P_{j_1}, P_j \neq 0, \text{ and } P_{j_1} \neq P_j.$$

It should be pointed out that P_{j_1} stands for any city placed in the first position of the sequence. Successive application of the previous recursive relation throughout the n stages leads to the optimal route such that

$$f_n(1\, P_{j_1}, \ldots, P_{j_{n-1}}) = \min_{m=1,\ldots,n-1} [f_{n-1}(P_{j_1},\ldots,P_{j_{m-1}},P_{j_{m+1}},\ldots,P_{j_{n-1}}) + C(P_{j_n}\, 1)].$$

Bellman [4] has reported on solutions to problems with up to 17 cities, exhausting the core capacity of the IBM 7094, by using a dynamic programming algorithm. Held and Karp [21] while working on sequencing problems have applied a dynamic programming algorithm for the traveling salesman problem.

Integer programming. In 1954, Dantzig and Fulkerson [11] have formulated the traveling salesman problem as a linear programming problem. The main characteristic of this formulation is that it involves a great number of variables and constraints that are required in the model. For an n-city problem with symmetric cost matrix, there exist $2^{n-1}-1$ loop constraints, n arrival constraints, n departure constraints, and $(n^2-n)/2$ integer-valued constrained variables. The approach used by Dantzig and Fulkerson consists of starting with only a few constraints, and then, adding the constraints required as the procedure progresses. Fractional solutions are disregarded by means of cutting-plane type constraints [18]. Miller, Tucker and Zemlin [27] have experienced with little success the cutting-plane algorithm. Ohter investigators [25, 11] have attempted to develop integer programming algorithms taking advantage of the particular structure of the problems solved; however, their approach cannot be used in general form. An efficient integer programming algorithm, specially for reasonable size

problems has been reported by Char [6].

The integer programming formulation of the traveling salesman problem is given next. To facilitate the exposition, the following changes in notation are made for this particular case such that

$$C_{ij} = C(i,j),$$

and

$$x_{ij} = A(i,j).$$

Thus, the objective function is such that

Minimize

$$\sum_{i=1}^{n} \sum_{j=1}^{n} C_{ij} x_{ij}; \qquad i \neq j;\ i, j = 1, 2, \ldots, n \tag{10}$$

subject to

$$\sum_{j=1}^{n} x_{ij} = 1; \qquad i = 1, 2, \ldots, n \tag{11}$$

$$\sum_{i=1}^{n} x_{ij} = 1; \qquad j = 1, 2, \ldots, n \tag{12}$$

$$x_{ij} = 1 \text{ or } 0$$

and the solution constitutes a tour, that is, subtours are eliminated. As an illustration, the complete integer formulation for a 4-city traveling salesman problem with non-symmetric distance matrix is as follows:

Minimize

$$Z = C_{12}\,x_{12} + C_{13}\,x_{13} + C_{14}\,x_{14} +$$
$$C_{21}\,x_{21} + C_{23}\,x_{23} + C_{24}\,x_{24} +$$
$$C_{31}\,x_{31} + C_{32}\,x_{32} + C_{34}\,x_{34} +$$
$$C_{41}\,x_{41} + C_{42}\,x_{42} + C_{43}\,x_{43}$$

Subject to

1. Arrival constraints

$$x_{12} + x_{13} + x_{14} = 1$$
$$x_{21} + x_{23} + x_{24} = 1$$
$$x_{31} + x_{32} + x_{34} = 1$$
$$x_{41} + x_{42} + x_{43} = 1,$$

2. Departure constraints:

$$x_{21} + x_{31} + x_{41} = 1$$
$$x_{12} + x_{32} + x_{42} = 1$$
$$x_{13} + x_{23} + x_{43} = 1$$
$$x_{14} + x_{24} + x_{34} = 1,$$

3. Subtour constraints:

$$x_{12} + x_{21} \leq 1$$
$$x_{13} + x_{31} \leq 1$$
$$x_{14} + x_{41} \leq 1$$
$$x_{23} + x_{32} \leq 1$$
$$x_{24} + x_{42} \leq 1$$

$$x_{12} + x_{23} + x_{31} \leq 2$$

$$x_{12} + x_{24} + x_{41} \leq 2$$

$$x_{13} + x_{32} + x_{21} \leq 2$$

$$x_{13} + x_{34} + x_{41} \leq 2$$

$$x_{14} + x_{42} + x_{21} \leq 2$$

$$x_{14} + x_{43} + x_{31} \leq 2$$

$$x_{23} + x_{34} + x_{42} \leq 2$$

$$x_{24} + x_{43} + x_{32} \leq 2$$

$$x_{34} + x_{42} + x_{23} \leq 2.$$

4. Integer-valued variable constraints:

$$x_{12} = x_{13} = x_{14} = x_{21} = x_{23} = x_{24} =$$

$$x_{31} = x_{32} = x_{34} = x_{41} = x_{42} = x_{43} = 0 \text{ or } 1.$$

Observation of the preceding mathematical model reveals the tremendous number of constraints and variables which have to be handled. The application of the model to problems having symmetric distance matrices results in reduced number of variables and constraints.

Branch-and-Bound. In 1963, Little et. al. [26] have developed an exact procedure for the solution of the traveling salesman problem using the enumerative approach based on the concepts established under the branch-and-bound heading in Section 1.1; thus, applying a tour building solution generating method. Eastman [14] and Shapiro [33] have developed branch-and-bound algorithms using subtour elimination schemes. Discussion of the branch-and-bound algorithm is given in detail in Section 2.1.

Heuristic algorithms for the solution of the traveling salesman problem comprise many approaches difficult to put under a few classes, so a discussion of the most important investigations follows.

A set of tour-to-tour improvement algorithms has been developed by Reiter and Sherman [31], designated as ALGO IV (r). The basic procedure of ALGO IV (1) consists of starting with a random tour, and then relocating the first city of this tour to find the sequence position that gives the minimal possible route length. This new tour is then modified by transposing its first city to the last sequence-position. Relocation of the first city in the last modified route is performed and the procedure is repeated until n sets of n-1 tours are examined without improvement in the value of the objective function. ALGO IV (2), ALGO IV (3), ..., ALGO IV (r) are similar procedures in which the search is performed for finding the best location for groups of 2, 3, ..., r cities, respectively in the current modified route such that the total route length is minimal. Lin [24] has reported on a set of algorithms similar to the ones of Reiter and Sherman, but, instead of relocating r cities in the sequence, he has found a better tour by replacing r links with r new links, the new route being referred to as r-opt tour [7]. A 2-opt tour is one that cannot be improved by replacing any 2 links in the tour by other two links. An n-opt tour is, of course, an optimal solution to the traveling salesman problem. A 2-opt tour is not necessarily the same as a 3-opt tour,..., or r-opt tour. Only when the distance matrix represents a convex poligon in the euclidean space, the 2 opt tour is equal to any other r-opt tour, and all of them are the same unique optimal solution for the given problem.

Reduction in the size of the problem under consideration is possible when a given subset S = {1, 2, ..., n-k} of cities has to be visited

consecutively in a preestablished order by considering this partial sequence as one synthesized link; thus if n-k cities are synthesized as city θ, the size of the problem is reduced from n to k+1, and the distance matrix C becomes

$$C'(i,j) = C(i,j), \qquad i,j \in S,$$

$$C'(\theta,j) = C(n-k,j), \qquad j \in S,$$

and

$$C'(i,\theta) = C(i,1), \qquad i \in S.$$

Held and Karp [20] and Karg and Thomson have reported some variations in the method of partitioning the traveling salesman problem, specially when the size of the problem is too large. Raymond [30] has developed a tour-to-tour improvement algorithm using an approach similar to that of Karg and Thomson. The first route is constructed using a tour building procedure. Two cities are first selected. Every other city is then tested against these two, and one is chosen for inclusion in the route such that the total route length is a minimum. The procedure continues until all cities have been included in the tour. A 3-opt procedure is then applied to obtain an improvement of the objective function.

1.3 Proposed Research

The purpose of this research is to study and develop a heuristic algorithm for the solution of the traveling salesman problem such that the computational effort is reduced in comparison with other existing algorithms. As it will be discussed in Chapter II, the basic ideas used are somewhat similar to those of the branch-and-bound approach. Heuristic

rules are developed in order to establish the assignment of links under the tour-building scheme.

Several problems from the current literature are to be solved and comparative evaluation has to be made against the optimal or best known solutions. A computer program is coded in FORTRAN IV for the IBM 360/50 computer. The computation time for obtaining the solution of each problem is reported. Details of the computer program are shown in Appendix A.

Some related experiments are performed and reported in Chapter III along with the computational experience of the proposed algorithm. Various ramifications to the heuristic rules are investigated and compared with the others developed.

CHAPTER II

DEVELOPMENT OF A HEURISTIC ALGORITHM

The proposed algorithm is somewhat similar to the branch-and-bound approach initially developed by Little et. al. [25] for the traveling salesman problem.

The main objective of this research is to limit the search for an optimal or near optimal solution for the traveling salesman problem by developing a heuristic algorithm using a look ahead rule. This method reduces the number of alternatives, and consequently, the amount of storage required by the computer to a manageable size.

This chapter is devoted to the development and illustration of the heuristic algorithm. This algorithm has been coded in FORTRAN IV for the IBM 360/50 computer using the WATFOR compiler. The listing of the computer program and solutions to various problems collected from the current literature are given in appendices A and B respectively.

2.1 Basic concepts

The solution of the traveling salesman problem involves the search for a route that provides the minimum total cost (or distance) among the feasible solutions for the given problem. This search can be performed by one of the methods mentioned in Chapter I. The proposed algorithm is somewhat similar to the branch-and-bound algorithm without backtracking developed by Ashour [2] for solving the flow-shop scheduling problem.

The search by the branch-and-bound approach involves the use of two basic ideas: (1) partitioning the set of possible solutions into succesively smaller subsets, and (2) imposing a criterion to help identify

a subset containing an optimal solution. Starting from the set of all possible routes, smaller subsets of routes are identified successively such that a certain criterion yields different values for the objective function according to the level at which the partitioning occurred. This procedure is continued until a minimum value is found. It is possible to find all optimal solutions by following the branching process. However, the computational effort and computer storage requirements are increased immensely. A reason for the enormous storage requirements is the back-tracking process employed after every decision has been made in order to discover if there is an unexplored node having a value of the objective function lower than that corresponding to the previous decision. Another reason is the occurrence of several sets of routes having the same value for the objective function. Thus, it is required to store all the un-explored nodes for further analysis.

Considering a n-city traveling salesman problem, the first node in the assignment tree is the ALL ROUTES node, which includes all the feasible routes for the given problem. At level 1, branching is made toward 2 nodes: (1) the node (I*, J*) formed by the set of all routes including link (I*, J*), and (2) the node $(\overline{I^*, J^*})$ containing the set of routes not including link (I*, J*). This branching process serves to construct the alternatives from which we find the optimal route, thus at level L, the assignment tree may have from L to 2^L nodes.

The bounding process involves the application of bounds to help eliminate some of the subsets obtained in the previous partitions. Hence, it improves the current solution successively until an optimal solution is found.

Since our concern in this thesis is to develop a computationally efficient algorithm, node $\overline{(I^*, J^*)}$ is discarded from further consideration. As a consequence, only one node at each level is selected by the application of the link selection criterion and the look ahead heuristic rule. The scheduling tree is then reduced to a single-branch tree and no backtracking is necessary.

Upon examining the distance (or cost) matrix of a given traveling salesman problem, it may have many links with zero value sufficient for constructing a route. Such a route would be an optimal solution. However, the distance matrix is generally given such that those zeros do not exist. In attempting to generate as many zeros as possible in the distance (or cost) matrix from which a final route is eventually formed, the reduction procedure can be applied to the distance matrix. As our problem is one of a sequential nature, the links having zero values should be given the highest priority for inclusion in the final route. Let (I, J) be a link having zero value in the current distance matrix. If link (I, J) is included in the final route, the total distance traveled is not affected because the corresponding value has been considered in the matrix reduction process. The alternative of not including node (I, J) requires the selection of two other links; one starting from city I and the other leading to city J. Thus, the cost of not including link (I, J) can be formed by the sum of the smallest value in row I plus the smallest value in column J. The sum of these two costs is referred to as the regret. Our criterion is to select the link which has the largest regret value for further investigation. Inclusion of link (I*, J*) having the largest regret in the final route implies that no further link can be assigned starting from city I* or ending

at city J*. Consequently, row I* and column J* are deleted from the distance matrix.

As the number of feasible solutions to the traveling salesman problem is n!, the number of solutions we have implicitly enumerated at level L is

$$N = n(n-1) \ldots (n-L),$$

and the number of solutions to be studied is

$$N' = n! - n(n-1) \ldots (n-L)$$

$$= n! \left(1 - \frac{1}{(n-L+1)}\right),$$

at stage L = n, all feasible solutions will have been exhausted.

The ties between the links having the largest regret in the selection procedure can be broken by a random selection or an application of a look ahead rule. The look ahead rule improves the possibilities of selecting the best candidate link. In this thesis we propose a look ahead rule such that we select the link which provides the lowest reduction of the distance matrix for the next level. Assuming that there exist more than one link having the same maximum regret at a certain level L, one of these links is temporarily selected to be included in the final route and the corresponding reduction of the distance matrix is computed. The procedure is repeated for each link, separately. Recalling now that if a route could be formed among the links with zero value in the distance matrix for the next level, such a route would be an optimal solution for the reduced matrix, then, the total length traveled by the salesman will be increased from level L to level L+1 by the amount of reduction undergone by the

distance matrix. Consequently, we propose the selection of the best candidate link such that

$$D(I^*, J^*) = \min_{(I,J)} \left\{ D(I, J) \right\}.$$

As a result of the decrease in the size of the distance matrix, due to the deletion of rows and columns involved in the decision making procedure after n-2 links have been included in the final route, what remains of the original matrix is a 2x2 reduced matrix. In the remaining matrix, there are only two candidate links having the same regret value; at this level it is not necessary to apply the look ahead rule because the reduction generated by either of the two links amounts to the same value, thus the tie must be broken according to any particular rule.

2.2 Sample problem

The proposed algorithm for the traveling salesman problem is illustrated by a sample problem. Consider a traveling salesman problem having 10 cities [1]. The corresponding distance matrix is given in table 2.1. Note that the distance matrix is non-symmetrical. We now proceed to the step-by-step solution of the sample problem.

Step 1. Initialize the link assignments. As we have made no assignments, we set level index L = 1 and assign

$$A(i,j) = 0, \qquad i, j = 1, 2, \ldots, 10.$$

Step 2. Compute the regrets. Reduction of the distance matrix is not required at this level for this particular problem because the original distance matrix was given in reduced form. The regrets are evaluated for

THIS BOOK CONTAINS NUMEROUS PAGES WITH DIAGRAMS THAT ARE CROOKED COMPARED TO THE REST OF THE INFORMATION ON THE PAGE.

THIS IS AS RECEIVED FROM CUSTOMER.

TABLE 2.1 Original distance matrix of a sample problem

FROM	1	2	3	4	5	6	7	8	9	10
1	∞	51	55	90	41	63	77	69	0	23
2	50	∞	0	64	8	53	0	46	73	72
3	30	77	∞	21	25	51	47	16	0	60
4	65	0	6	∞	2	9	17	5	26	42
5	0	94	0	5	∞	0	41	31	59	48
6	79	65	0	0	15	∞	17	47	32	43
7	76	96	48	27	34	0	∞	0	25	0
8	0	17	9	27	46	15	84	∞	0	24
9	56	7	45	39	0	93	67	79	∞	38
10	30	0	42	56	49	77	76	49	23	∞

all cells having zeros in the reduced matrix. For example, consider link (1, 9). The regret corresponding to this link is computed such that

$$R(1, 9) = \min_{\substack{k \\ k \neq 9}} [C(1, k)] + \min_{\substack{k \\ k \neq 1}} [C(k, 9)]$$

$$= \min [\infty, 51, 55, 90, 41, 63, 77, 69, 23]$$

$$+ \min[73, 0, 26, 59, 32, 25, 0, \infty, 23]$$

$$= 23.$$

The resulting regrets of the zero cells appear in the upper left hand corner of cells in table 2.2.

Step 3. Select the link (I*, J*) for possible inclusion in the final route. The matrix depicted in table 2.2 shows that links (1, 9), (7, 10), and (10, 2) have the largest regret value which is 23. Now we apply the look ahead rule to select the link which will generate the lowest total reduction in the distance matrix in case of being included in the final route. First, we temporarily include link (1, 9) in the final route. The resulting reduced matrix after deletion of row 1 and column 9 will undergo a reduction of 16 units. Similarly, if we introduce, one at a time, links (7, 10) and (10, 2) in the final route, reductions of 5 and 2 units, respectively, occur in the distance matrix. The tie is then broken in favor of link (10, 2) generating the lowest reduction. Thus link (10, 2) is selected as the best candidate link.

Step 4. Check the level index. As L = 1 and the selected link does not close a loop we include it in the final route by setting the assignment

TABLE 2.2 Reduced distance matrix and regrets for L = 1

FROM \ TO	1	2	3	4	5	6	7	8	9	10
1	∞	51	55	90	41	63	77	69	23 0	23
2	50	∞	0 0	64	8	53	17 0	46	73	72
3	30	77	∞	21	25	51	47	16	16 0	60
4	65	2 0	6	∞	2	9	17	5	26	42
5	0 0	94	0 0	5	∞	0 0	41	31	59	48
6	79	65	0 0	5 0	15	∞	17	47	32	43
7	76	96	48	27	34	0 0	∞	5 0	25	23 0
8	0 0	17	9	27	46	15	84	∞	0 0	24
9	56	7	45	39	9 0	93	67	79	∞	38
10	30	23 0	42	56	49	77	76	49	23	∞

A(10, 2) = 1. Row 10 and column 2 are deleted from the distance matrix by setting

$$c(10, k) = c(k, 2) = \infty, \qquad k = 1, 2, \ldots, 10.$$

and

$$c(2, 10) = \infty.$$

The index level is increased by 1 to become 2. As L < n or 2 < 10 we proceed to step 2.

Step 2. Compute the regrets. The reduced distance matrix and corresponding regrets are shown in table 2.3.

Step 3. Select a link (I*, J*) for possible inclusion in the final route. In the matrix of table 2.3, link (9, 5) has the largest regret of 38. Thus, link (9, 5) is selected.

Step 4. Check the level index. As L = 2 and the selected link does not close a loop, that is, links (10, 2) and (9, 5) do not form a loop, we include link (9, 5) in the final route by setting A(9, 5) = 1. Row 9 and column 5 are deleted from the distance matrix such that

$$c(9, k) = c(k, 5) \infty, \qquad k = 1, 2, \ldots, 10,$$

and

$$c(5, 9) = \infty.$$

The index level is increased by 1 to become 3. As L < n or 3 < 10 we proceed to step 2.

TABLE 2.3 Reduced distance matrix and regrets for L = 2

FROM \ TO	1	3	4	5	6	7	8	9	10
1	∞	55	90	41	63	77	69	23 0	23
2	50	0 0	64	8	53	15 0	46	73	∞
3	30	∞	21	25	51	47	16	16 0	60
4	63	4	∞	3 0	7	15	3	24	40
5	0 0	0 0	5	∞	0 0	41	31	59	48
6	79	0 0	5 0	15	∞	17	47	32	43
7	76	48	27	34	0 0	∞	3 0	25	23 0
8	0 0	9	27	46	15	84	∞	0 0	24
9	56	45	39	38 0	93	67	79	∞	38

Step 2. Compute the regrets. The reduced matrix and corresponding regrets are shown in Table 2.4.

Step 3. Select the link (I*, J*) for possible inclusion in the final route. The matrix in Table 2.4 shows that links (1, 9) and (7, 10) have the largest regret value which is 23. Applying the look ahead rule, the reduction generated by links (1, 9) and (7, 10), separately, are 16 and 0 respectively. Thus link (7, 10) is selected.

Step 4. Check the level index. As L = 3 and the selected link does not close a loop, that is, links (10, 2), (9, 5) and (7, 10) do not form a loop, link (7, 10) is included in the final route by setting A(7, 10) = 1. Row 7 and column 10 are deleted from the distance matrix such that

$$C(7, k) = C(k, 10) = \infty, \qquad k = 1, 2, \ldots, 10,$$

and

$$C(10, 7) = \infty.$$

The index level is increased by 1 to become 4. As L < n or 4 < 10 we proceed to step 2.

Step 2. Compute the regrets. The reduced distance matrix and corresponding regrets are shown in Table 2.5.

Step 3. Select a link, (I*, J*) for possible inclusion in the final route. In the matrix of Table 2.5, link (1, 9) has the largest regret of 55. Thus link (1, 9) is selected.

TABLE 2.4 Reduced distance matrix and regrets for L = 3

FROM \ TO	1	3	4	6	7	8	9	10
1	∞	55	90	63	77	69	23 0	23
2	50	0 0	64	53	12 0	46	73	∞
3	30	∞	21	51	47	16	16 0	60
4	60	1	∞	4	12	1 0	21	37
5	0 0	0 0	5	0 0	41	31	∞	48
6	79	0 0	5 0	∞	17	47	32	43
7	76	48	27	0 0	∞	0 0	25	23 0
8	0 0	9	27	15	84	∞	0 0	24

Step 4. Check the level index. As L = 4 and the selected link does not close a loop, that is, links (10, 2), (9, 5), (7, 10), and (1, 9) do not close a loop, link (1, 9) is included in the final route by setting A(1, 9) = 1. Row 1 and column 9 are deleted from the distance matrix such that

$$C(1, k) = C(k, 9), \qquad k = 1, 2, \ldots, 10,$$

and

$$C(9, 1) = \infty.$$

The index level is increased by 1 to become 5. As L < n or 5 < 10 we proceed to step 2.

Step 2. Compute the regrets. The reduced distance matrix and corresponding regrets are shown in Table 2.6.

Step 3. Select a link (I*, J*) for possible inclusion in the final route. Link (2, 7) having the largest of 12 is selected.

Step 4. Check the level index. As L = 5 and the selected link does close a loop, that is, links (10, 2), (9, 5), (7, 10), (1, 9), and (2, 7) form the loop (10, 2) → (2, 7) → (7, 10), link (2, 7) is deleted by setting C(2, 7) = ∞, and then we proceed to step 3.

Step 3. Select a link (I*, J*) for possible inclusion in the final route. From table 2.6 link (8, 1) with the largest regret of 9 is selected.

Step 4. Check the level index. As L = 5 and the selected link does not close a loop, link (8, 1) is included in the final route by setting

TABLE 2.5 Reduced distance matrix and regrets for L = 4

FROM \ TO	1	3	4	6	7	8	9
1	∞	55	90	63	77	69	55 0
2	50	0 0	64	53	12 0	46	73
3	30	∞	21	51	47	16	16 0
4	60	1	∞	4	12	17 0	21
5	0 0	0 0	5	4 0	41	31	∞
6	79	0 0	5 0	∞	17	47	32
8	0 0	9	27	15	84	∞	0 0

TABLE 2.6 Reduced distance matrix and regrets for L = 5

FROM \ TO	1	3	4	6	7	8
2	50	0 0	64	56	12 0	46
3	14	∞	5	35	31	5 0
4	60	1	∞	4	12	1 0
5	0 0	0 0	5	4 0	41	31
6	79	0	5 0	∞	17	47
8	9 0	9	27	15	84	∞

(8, 1) = 1. Row 8 and column 1 are deleted from the distance matrix such hat

$$C(8, k) = C(k, 1) = \infty, \qquad k = 1, 2, \ldots, 10,$$

nd

$$C(1, 8) = \infty.$$

he index level is increased by 1 to become 6. As L < n or 6 < 10 we roceed to step 2.

The problem solving procedure continues smoothly through levels 6-9, here links (2, 3), (6, 4), (5, 6), and (3, 8), respectively, are included n the final route. Tables 2.7 through 2.10 illustrate the respective istance matrices and corresponding regrets. After increasing the index evel from 9 to 10 we proceed to step 2.

Step 2. Evaluate the regrets. The distance matrix is now 1 x 1 and he regret for the only remaining link, (3, 8), is ∞.

Step 3. Select link (I*, J*) for possible inclusion in the final oute. Thus, link (3, 8) is selected.

Step 4. Check the level index. As L = n or 10, and the selected link oes close a loop, link (3, 8) is included in the final route by setting (3, 8) = 1.

Step 5. Evaluate the final route. Upon examining the cells which ave the assignment such that A(I*, J*) = 1, the following sequence of

TABLE 2.7 Reduced matrix and regrets for L = 6

FROM \ TO	3	4	6	7	8
2	46 0	64	53	∞	46
3	∞	5	35	19	5 0
4	1	∞	4	5 0	0 0
5	0 0	5	4 0	29	31
6	0 0	5 0	∞	5	47

TABLE 2.8 Reduced matrix and regrets for L = 7

FROM \ TO	4	6	7	8
3	5	35	19	5 0
4	∞	4	5 0	0 0
5	5	9 0	29	31
6	10 0	∞	5	47

TABLE 2.9 Reduced matrix and regrests for L = 8

FROM \ TO	6	7	8
3	35	19	19 0
4	∞	19 0	0 0
5	64 0	29	31

TABLE 2.10 Reduced matrix and regrets for L = 9

FROM \ TO	7	8
3	19	19 0
4	19 0	0 0

cities is formed, considering city number 1 as the depot:

$$1 \to 9 \to 5 \to 6 \to 4 \to 7 \to 10 \to 2 \to 3 \to 8 \to 1.$$

Finally, the total distance traveled for the above final route is:

$$\begin{aligned} C &= C(1,9) + C(9,5) + C(5,6) + C(6,4) + C(4,7) + C(7,10) + C(10,2) + C(2,3) \\ &\quad + C(3,8) + C(8,1) \\ &= 0 + 0 + 0 + 0 + 17 + 0 + 0 + 0 + 16 + 0 \\ &= 33, \end{aligned}$$

where the respective distances are those given in the original distance matrix.

It is of interest to note that the above solution is an optimal route, since it is obtained by Ackoff [1] when a branch-and-bound algorithm has been used.

2.3 Computational algorithm.

In summary, the heuristic algorithm is stated step by step as follows:

Step 1. Initialize the city link assignments.

1.1 Set the level index L = 1

1.2 Set all the link assignments

$$A(i,j) = 0, \qquad i, j = 1, 2, \ldots, n.$$

Step 2. Compute the regrets.

2.1 Reduce the values in the matrix by reducing the rows such that for each row i,

$$C(i,j) = C(i,j) - \min_{j} \left[C(i,j) \right], \quad j = 1, 2, \ldots, n,$$

and then reducing the columns of the resulting matrix such that for each column j,

$$C(i,j) = C(i,j) - \min_{i} \left[C(i,j) \right], \qquad i = 1, 2, \ldots, n.$$

2.2 Evaluate the regrets for all cells having zeros in the reduced matrix such that

$$R(I,J) = \min_{\substack{k \\ k \neq J}} \left[C(I,k) \right] + \min_{\substack{k \\ k \neq I}} \left[C(k,J) \right].$$

Step 3. Select the link (I*, J*) for possible inclusion in the final route such that

$$R(I^*, J^*) = \max_{(I,J)} \left[R(I,J) \right],$$

3.1 If a tie does not exist, go to step 4.

3.2 If a tie exists and L < n-1, delete temporarily the row and column associated with each tied link, one at a time, and reduce the values in the remaining rows first and columns next. Then, add the amount of reductions such that

$$D(I^*, J^*) = \min_{j} \left[C(i,j) \right] + \min_{i} \left[C(i,j) \right].$$

Select the link having the minimum amount of reduction. If a tie still exists, break it by any particular rule.

3.3 If a tie exists and L = n-1, break the tie by any particular rule.

Step 4. Check the level index.

4.1 If $L < n$ and the selected link (I^*, J^*) does not close a loop, include it in the final route by setting

$$A(I^*, J^*) = 1,$$

and delete the corresponding row and column by setting

$$C(I^*, k) = C(k, J^*) = \infty, \qquad k = 1, 2, \ldots, n,$$

and

$$C(J^*, I^*) = \infty.$$

Then, set $L = L+1$ and go to step 2.

4.2 If $L < n$ and the selected link closes a loop, delete it from the distance matrix by setting

$$C(I^*, J^*) = \infty,$$

and go to step 3.

4.3 If $L = n$, include the selected link in the final route by setting

$$A(I^*, J^*) = 1$$

and go to step 5.

Step 5. Evaluate the final route.

5.1 Set the sequence of links having

$$A(I^*, J^*) = 1.$$

5.2 Find the total cost (or distance) such that

$$C = \sum_{(I^*,J^*)} C(I^*,J^*).$$

The solution will result in a complete route with the minimum possible total distance or cost. It should be pointed out; however, that this solution may or may not be the optimal one.

CHAPTER III

EXPERIMENTAL INVESTIGATION

This chapter is composed of five sections. Section 3.1 includes the experience obtained in solving various problems using the proposed algorithm. Section 3.2 is devoted to some ramifications of the proposed algorithm and the corresponding results. The remaining sections include the results of several related experiments performed in order to evaluate the application and possibilities of new procedures for the solution of the traveling salesman problem.

3.1 Computational results

Results obtained using the heuristic algorithm developed in Chapter II are reported in this section. All problems solved have been taken from the literature so as to make a comparison with results previously obtained. Original methods of solution for problems presented here are quite varied. It should be pointed out; however, that the comparison between the computation time required by the various algorithms are difficult to make.

One of the features of the proposed algorithm and its computer program is the ability to handle problems having symmetric and nonsymmetric distance matrices by using the same approach and without change in notation.

In regard to storage requirements, consideration of the number of cells involved in the input matrix points out that for problems having symmetric distance matrices, we could work with only half of the matrix, thus saving a great deal of memory in the computer. This relative disadvantage has been overcome mainly because during the branching process only one node is saved at each level. Furthermore, the number of assignment variables is

n^2 for all cases and this number cannot be reduced when working with problems having nonsymmetric distance matrices. The algorithm itself and the computer problem presented in this thesis have been implemented in such a way that both types of problems, symmetric and nonsymmetric, can be handled with the same relative success. There is one difference at the input stage, however. For nonsymmetric matrices it is only required to consider those city links located below the main diagonal of the distance matrix. The computer program will take care of completing the matrix. Thus $(n^2-n)/2$ values are introduced and the program completed to n^2-n values, taking advantage of the relation

$$C(J,I) = C(I,J),$$

and completion of the distance matrix is achieved by giving a value of ∞ to cells which involve traveling from city I to itself, for all I; and for nonsymmetric distance matrices n^2 values have to be introduced. The foregoing procedure allowed us some simplification in the preparation of the data.

One of the characteristics of the traveling salesman problem that makes it difficult to solve is the need for eliminating incomplete loops or subtours during the problem solving procedure. These subtours must be prevented, and, as shown by Char [6], they give rise to many additional constraints imposed on the problem, specially when working with the 0 - 1 integer programming type algorithms. The device to avoid such subtours implemented in this research is to eliminate each link from further consideration whenever it closes the loop before visiting all the cities. That is, if the current procedure is at level L and the selected link

closed the loop, it is deleted from the distance matrix by setting its value in the matrix equal to infinity. Then, the link with the next highest regret value is searched. This method constitutes a simple and expeditive way of imposing explicitly the subtour constraints and it is implemented in the computer program by means of the BLOCK subroutine.

The results of the application of the heuristic algorithm are encouraging, specially regarding the quality of the solution and the relatively small computation time required for solution on the IBM 360/50 computer. Fourteen problems were solved with the heuristic algorithm, the number of cities ranging from 4 to 42. Most of the problems studied were of small and medium size and only one large scale problem have been solved. Out of the fourteen problems solved, optimal solutions have been found for eight. The optimal solution is not available for problems 7 and 10, but the routes obtained for them are equal to the best known for problem 7 and an improved solution for problem 10. In this case, the efficiency of the solution has to be taken with reference to the best known which constitutes a very optimistic approach. The solutions of the problems 2, 4, 9, and 14 resulted in suboptimal routes with efficiencies of 0.95, 0.95, 0.97, and 0.85. The results of the application of the proposed algorithm are given in complete detail in Appendix B, including the original data. A summary is presented in Table 3.1.

It is worth while to mention that the application of the present algorithm for the traveling salesman problem terminates at the completion of only one route. Thus, in those cases where there exist various routes having the minimal distance, it is restricted to identify only one of them. Although all optimal solutions are mathematically equivalent, it is sometimes advantageous, from a realistic point of view, to know the existence of all of them, if any.

TABLE 3.1 Summary of results

PROBLEM NUMBER	REF-ERE-NCE	NO. OF CITIES	DISTANCE MATRIX	OPTIMAL(*) OR BEST KNOWN SOLUTION				SOLUTION WITH PROPOSED ALGORITHM		
				METHOD USED	ROUTE LENGTH	COMPUTER TYPE	COMPUTER TIME (SEC.)	ROUTE LENGTH	TIME (SEC.)	EFFICIENCY OF SOLUTION
1	[6]	4	NONSYM	0-1 IP	11*	IBM 360/50	4.00	11*	2.01	1.00
2	[29]	5	NONSYM	Heuristic	20*			21	3.09	.95
3	[32]	5	NONSYM	B and B	15*			15*	3.30	1.00
4	[36]	5	NONSYM	B and B	62*			65	2.62	.95
5	[26]	6	NONSYM	B and B	63*			63*	5.28	1.00
6	[30]	7	SYM	Heuristic	179*			179*	6.81	1.00
7	[1]	10	NONSYM	B and B	33*			33*	17.96	1.00
8	[28]	6	SYM	NETWORK	37			37	4.80	1.00
9	[23]	5	SYM	Heuristic	148*			152	4.20	.97
10	[28]	5	SYM	NETOWRK	38			37	2.87	Improved
11	[23]	10	SYM	Heuristic	378*			378*	14.62	1.00
12	[10]	6	NONSYM	B and B	20*			20*	4.00	1.00
13	[37]	5	SYM	B and B	6097*			6097*	2.46	1.00
14	[12]	42	SYM	Heuristic	699	BDXG-20	270	802	273.35	.85

3.2 Ramifications of the algorithm

The objective of improving the quality of solution has led us to consider the possibilities of applying a look ahead rule of two levels in depth. It has been suspected that with such a rule, the ties will be broken advantageously in the selection of the best candidate link (I*,J*) for inclusion in the final route. This approach has been used resulting in an increased computational effort and no apparent improvement in the quality of solution. The two-depth look ahead rule has been applied to only those problems which did not respond optimally to the basic heuristic algorithm proposed in Chapter II.

The two-depth look ahead rule has been applied when the tie of links having the largest regret at level L, has led to another tie at level L+1 in terms of the total amount of reduction of the distance matrix generated by the temporary inclusion of the tied links at levels L and L+1. The best candidate link is then selected such that in case of being assigned in the final route it will generate the smallest reduction of the matrix before level L+2 is investigated.

A consequence of the two-depth look ahead rule is that it increases the number of alternatives to be explored. This causes an expected increase in the computational effort. Some efficient and powerful rules for the two-depth look ahead approach can be developed such as the one devised by Hering [22] for the solution of the delivery problem.

3.3 Application of the saving approach

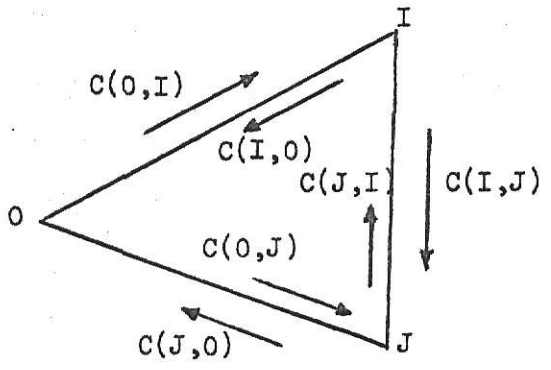
The traveling salesman problem can be treated as a special case of the delivery problem. The delivery problem consists of determining a set

of routes to deliver goods from a central depot to several demand points such that certain capacity and load restrictions of the carriers are not violated. The traveling salesman problem can then be thought of as a delivery problem in which the only carrier available, the salesman, has a capacity large enough to supply all the customers in only one trip and there is no restriction on the length of this unique route. The existence of the central depot is not important in the search for the solution of the traveling salesman problem because the solution consists of a single loop, and any of the cities can be considered as the depot. However, in an attempt to apply the saving approach and work an example, let us assume that city 1 constitutes the central depot.

The savings approach involves two basic ideas: (1) assignment of one round trip from the depot to every demand point, and (2) elimination of links starting from and leading to the depot by joining demand points on the route such that the savings in distances gained by this procedure is maximized, thus minimizing the objective function, or the total distance traveled. Then, if cities I and J are on the route, and if the salesman has previously been assigned the routes 1-I-1 and 1-J-1, the savings resulting from linking I with J are such that

$$S_1(I,J) + C(I,1) + C(1,J) - C(I,J),$$

where C represents the cost or distance associated with the respective links, and S(I,J) is the savings, with respect to city 1, corresponding to the linking of cities I and J, as shown below



The savings approach [8,34] employed to develop heuristic *algorithms* for delivery problems is based on a rule to link those cities which maximize the savings such that

$$S(I^*,J^*) = \max_{(i,j)} \left[S(i,j) \right].$$

This selection procedure is carried out taking into consideration the constraints imposed on the system, step by step, until all demand points have been assigned in their respective routes. The only problems whose solution has been published using the savings approach are those with symmetric distance (or cost) matrices. Hence, in order to investigate its general application and possibilities, the formula for computing the savings has to be respected in the direction of the links according to the preceding figure. This provision enables the formula to be applied to both symmetric and nonsymmetric distance matrices. The sample problem illustrated in Chapter II is solved using the savings approach. Table 3.2 shows the savings corresponding to the elements of the matrix given for this sample problem. The selection of links, based on the highest current savings and the avoidance of closing loops before connecting all cities, leads to the successive assignment of the following links in the final route:

TABLE 3.2 Savings for the 10 - city traveling salesman problem

	TO									
FROM	1	2	3	4	5	6	7	8	9	10
1	$-\infty$	0	0	0	0	0	0	0	0	0
2	0	$-\infty$	105	76	83	60	127	73	-23	1
3	0	4	$-\infty$	99	46	42	60	83	30	-7
4	0	116	114	$-\infty$	104	119	125	129	39	46
5	0	-43	55	85	$-\infty$	63	36	38	-59	-25
6	0	65	134	169	105	$-\infty$	139	101	47	59
7	0	31	83	139	83	139	$-\infty$	145	51	99
8	0	34	46	63	-5	48	-7	$-\infty$	0	-1
9	0	100	66	107	97	26	66	46	$-\infty$	41
10	0	81	43	64	22	16	31	50	7	$-\infty$

(6,4), (7,8), (2,7), (4,2), (9,5), (5,6), (8,3), (10,9), (1,10), and (3,1), with the following total traveled distance:

$$C = C(1,10) + C(10,9) + C(9,5) + C(5,6) + C(6,4) + (C(4,2) + C(2,7) + C(7,8) + C(8,3) + C(3,1)$$
$$= 23 + 23 + 0 + 0 + 0 + 0 + 0 + 0 + 9 + 30$$
$$= 85,$$

where the costs are those given in the original distance matrix of Table 2.1.

The result obtained does not correspond to the optimal solution of the problem. The following reasons seem to operate against the saving approach in comparison with the regret approach:

1. The savings are fixed in value along the problem solving procedure. There is no saving update for the remaining links after every decision has been made. This is not the case when regrets are employed.
2. The existence of the depot as a fixed location for reference is too restrictive to evaluate the savings. Evaluation of regrets is done with respect to all other cities.
3. It is required to device some complementary or look ahead rules before selecting the link for inclusion in the final route in order to obtain improved solutions.
4. Theoretically, in regard to the traveling salesman problem, there exists one different solution for every city used as central depot when savings are applied. The same cannot be said with regard to the regrets.

Observations of the previous experiments have led to the conclusion that there is no relation between the savings and regrets as a decision criterion

in the assignment of priorities for the selection of candidate links.

3.4 Modified Regret Formula

The use of the regret as applied only to the links having zero value in the current distance matrix seemed to be an imperfect method of selection of the candidate links. For instance, if we consider any link, (i,j) in the distance matrix, its regret could be conceived as the cost of not having the smallest cost in row i plus the cost of the cell having the smallest cost in column j minus the cost of link (i,j) itself, or simply,

$$R'(i,j) = \min_{\substack{k \\ k \neq j}} \left[C(i,k) \right] + \min_{\substack{k \\ k \neq i}} \left[C(k,j) \right] - C(i,j).$$

This relation can be applied to all the elements in the distance matrix and the modified regret used as a decision value for the selection criterion. Several problems have been solved using this modified formula. As expected, the computational effort has been increased enormously and the quality of solutions are certainly poor. A closer look at the modified regret formula for the traveling salesman problem shows that it should not be used for links other than zero for the following reasons:

1. Assignment of a zero link will not increase the total distance traveled after the last reduction,
2. Modified regrets for links not having a zero value are the negative of this value.

Consider link (i,j) whose value is different from zero in the reduced matrix, then

$$R'(i,j) = \min_{\substack{k \\ k \neq j}} \left[C(i,k)\right] + \min_{\substack{k \\ k \neq i}} \left[C(k,j)\right] - C(i,j)$$

However, it is necessary that

$$\min_{\substack{k \\ k \neq j}} \left[C(i,k)\right] = \min_{\substack{k \\ k \neq i}} \left[C(k,j)\right] = 0.$$

Thus,

$$R'(i,j) = - C(i,j).$$

Since the negative regrets are of no use for the traveling salesman problem, further study of the modified regret formula has been terminated. It is worthwhile to mention that the modified regret formula can be investigated in regard to some combinatorial problems in which the device of reducing the matrix is not applicable.

3.5 Assignment of zeros approach.

As mentioned in Chapter II, the assignment of links having zero value in the reduced distance matrix should lead to the discovery of an optimal solution provided that there are enough zeros as to form a route. This procedure has been attempted with one important variation. After exhaustion of the assignable zero links of the reduced matrix, a new reduction of the matrix has to be performed if the assigned links do not form the complete route. Assignment of priority is in favor of the link having the largest regret in the current distance matrix, provided it does not close the loop before visiting all cities. The application of this procedure has provided solutions with varied efficiency due to the fact that the assignment of

TABLE 3.3 First reduced matrix and corresponding regrets

FROM \ TO	1	2	3	4	5	6	7	8	9	10
1	∞	51	55	90	41	63	77	69	23 0	23
2	50	∞	0 0	64	8	53	17 0	46	73	72
3	30	77	∞	21	25	51	47	16	16 0	60
4	65	2 0	6	∞	2	9	17	5	26	42
5	0 0	94	0 0	5	∞	0 0	41	31	59	48
6	79	65	0 0	5 0	15	∞	17	47	32	43
7	76	96	48	27	34	0 0	∞	5 0	25	23 0
8	0 0	17	9	27	46	15	84	∞	0 0	24
9	56	7	45	39	9 0	93	67	79	∞	38
10	30	23 0	42	56	49	77	76	49	23	∞

zero cells in some problems caused deviation from the optimal route. However, optimal solutions have been obtained for some problems. This procedure is illustrated for the sample problem given in Chapter II. The first reduced distance matrix is given in Table 3.3. Selection of the best candidate link is done in favor of the one having the largest regret. In the case of a tie, the link with the largest J subscript is selected. The following links are successively assigned: (10,2), (7,10), (1,9), (9,5), (6,4), (8,1), (5,6), and (2,3). The deletion of respective rows and columns made during the previous assignments has resulted in the following distance matrix.

FROM \ TO	7	8
3	47	16
4	17	5

As there are no links with zero value, this matrix is reduced to

FROM \ TO	7	8
3	19	0
4	0	0

where the amount of reduction is 33. Continuing the assignment of links having zero value, links (4,7) and (3,8) are included in the final route. As the assignments are completed, the total distance traveled is 33. Thus

the value of the objective function, 33, obtained is the same as the one found for the same sample problem in Chapter II.

CHAPTER IV

SUMMARY AND CONCLUSIONS

Combinatorial problems consist of finding, from among a finite set of alternatives, one solution that optimizes some objective function, provided that the imposed constraints are not violated. The search of the optimal solution to combinatorial problems usually requires consideration of a great number of alternatives, which in turn makes the computational effort very large.

The traveling salesman problem consists of finding the minimal distance route starting and ending at the same terminal and visiting, once and only once, each of a number of cities.

The basic objective of the research on the traveling salesman problem presented in this thesis was to investigate some avenues leading to the development of optimal or near optimal solution procedures, aiming at the reduction of the computational effort involved and storage requirements when using the computer.

The traveling salesman problem can be considered as a special case of the delivery problem which consists of finding a set of routes, all starting and ending at the same terminal in order to satisfy the demands of a given set of customers, meeting imposed constraints, such the objective function, as the total distance traveled is minimized.

A study has been made of the various methods available for the solution of the delivery and traveling salesman problems. The common characteristic of these solution procedures is the existence of three main steps in the solution, namely, starting point, solution generation scheme, and termination

rule. A heuristic algorithm has been developed for the solution of the traveling salesman problem, where the starting point is the reduced original distance matrix. The solution generation scheme is one of tour building. Termination occurs when all cities have been assigned in the route. The algorithm makes use of the basic ideas of the branch-and-bound technique without backtracking. Successive assignment of links in the final route are made according to heuristic rules involving the use of regrets of links with zero value in the current distance matrix and the application of a look ahead procedure to break ties. At each level, the link having the largest regret is considered to be the best candidate for inclusion in the route. The look ahead rule stems from the reduction generated in the distance matrix by the provisional inclusion of each tied candidate link; ties are broken in favor of the link producing the smallest reduction.

The proposed algorithm has been coded in FORTRAN IV for the IBM 360/50 computer. Fourteen problems have been solved. The suggested approach looks promising because of the relative success obtained, specially for small and medium sized problems where high quality of solutions was obtained.

Some attempts were made to develop two-depth look ahead rules in order to improve the possibilities of reaching the optimal solution. Also a study of the savings approach was made in comparison with the regrets as decision criteria for the assignment of links in the route.

The following conclusions can be drawn:

1. Optimal solution searching procedures for combinatorial problems are limited in their application to problems of small size

because of the great number of variables and constraints involved. Even when the mathematical model is relatively easy to set, the complete enumeration of variables and constraints shows the difficulties to be met with.

2. The proposed algorithm for the traveling salesman problem works equally well on problems with symmetric or non-symmetric distance matrices. For the symmetric case, the solution corresponds to a route that can be traveled in any of both directions. Directionality of traveling is critical for problems with non-symmetrical distance matrices because of the different possible cost for traveling each link in any of the two directions.

3. The application of the savings approach to traveling salesman problems proved to be unsuccessful. The main setback of this approach rests on the static nature of the savings, which maintain the same value for all the non-assigned links throughout the tour building procedure.

4. There is no similarity between the savings and regrets as decision criteria for the assignment of priorities in the route building procedure for the traveling salesman problem.

5. Further investigation on the possibilities of the savings approach will serve to evaluate its applicability to optimizing routing problems. For the traveling salesman problem, in particular, it could investigate the development of methods using a more dynamic kind of savings, that is, taking into consideration not only one fixed city-terminal, but all of the demand points to be visited.

6. Two-depth look ahead rules demand investigation of an increased number of alternatives. The application of heuristic look ahead rules results in a more exhaustive, yet imperfect, enumeration of alternatives, causing an increase in the computational effort.

REFERENCES

1. Ackoff, R. L., and M. W., Sasieni, Fundamentals of Operations Research, John Wiley and Sons Inc., New York, 1968.

2. Ashour, S., "A Branch-and-Bound Algorithm for Flow Shop Scheduling Problems," Unpublished Working Papers, Kansas State University, Manhattan, Kansas (1970).

3. Ashour, S., Introduction to Scheduling: Concepts, Analysis, and Performances, John Wiley and Sons, New York, N.Y., in press.

4. Bellman, R., "Dynamic Programming Treatment of the Travelling Salesman Problem," Association for Computing Machinery, Vol. 9, 61-63, (1962).

5. Bellmore, M., and G. L. Memhauser, "The Traveling Salesman Problem: A Survey," Operations Research, Vol. 16, 538-558 (1968).

6. Char, A., "Application of Linear Pseudo-Boolean Programming to Combinatorial Problems," Master's Thesis, Kansas State University, Manhattan, Kansas (1969).

7. Christofedes, N. and S. Eilon, "An Algorithm for the Vehicle Dispatching Problem," Operations Research Quarterly, Vol. 20, No. 3, (1968).

8. Clarke, G., and J. W. Wright, "Scheduling of Vehicles from a Central Depot to a Number of Delivery Points," Operations Research, Vol. 12, No. 4, 568-581 (1964).

9. Cochran, H., "Optimization of a Carrier Routing Problem," Master's Thesis, Kansas State University, Manhattan, Kansas (1966).

10. Conway, R. N., W. L. Maxwell and L. W. Miller, Theory of Scheduling, Addison-Wesley Publishing Company, Reading, Massachusetts, 1967.

11. Dantzig, G. B., D. R. Fulkerson, and S. M. Johnson, "On a Linear Programming Combinatorial Approach to the Traveling-Salesman Problem," Operations Research, Vol. 7, 58-66 (1959).

12. _____, "Solution of a Large Scale Traveling Salesman Problem," Operations Research, Vol. 2, 393-410 (1954).

13. Dantzig, G. B., and J. H. Ramser, "The Truck Dispatching Problem," Management Science, Vol. 6, No. 1, 80-91 (1959).

14. Eastman, W. L., "Linear Programming with Pattern Constraints," Ph.D. Dissertation, Cambridge, Mass., Harvard University, (July 1958).

15. Elson, D. G., and E. Helloman, "Techniques for Truck and Driver Scheduling," Unpublished Working Papers-CEIR, 1-58 (June 1968).

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

17. Gaskell, T. J., "Bases for Vehicle Fleet Scheduling," Operations Research Quarterly, Vol. 18, No. 3, 281-295 (1967).

18. Gomory, R. E., "An Algorithm for Integer Solutions to Linear Programs," pp 269-302 in Recent Advances in Mathematical Programming, Robert L. Graves and Philip Wolfe (eds.), McGraw Hill Book Co., New York, 1963.

19. Gonzales, R. H., "Solution of the Traveling Salesman Problem by Dynamic Programming on the Hypercube", Interin, Technical Deport No. 18, Operations Research Center, Massachusetts Institute of Technology, Cambridge, Massachusetts (1962).

20. Hayes, R. L., "The Delivery Problem," Ph.D. Dissertation, Carnegie Institute of Technology (1967).

21. Held, M., and R. Karp, "A Dynamic Programming Approach to Sequencing Problems," SIAM 10, 196-210 (1962).

22. Hering, R. W., "Evaluation of Some Heuristic Look Ahead Rules for Multiple Terminal Delivery Problems," Master's Thesis, Kansas State University, Manhattan, Kansas (1970).

23. Karg, R. L., and G. L. Thompson, "A Heuristic Approach to Solving Traveling Salesman Problems," Management Science, Vol. 10, 225-248 (1964).

24. Lin, S., "Computer Solutions of the Traveling Salesman Problem," The Bell Technical Journal, Vol. 44, No. 10, 2245-2269 (1965).

25. Little, J. D. C., "Branch-and-Bound Methods for Combinatorial Problems," pp 118-133 in Operations Research and the Design of Management Information Systems, John F. Perce Jr. (ed.) Badger Printing Corporation, Appleton, Wisconsin, 1967.

26. Little, J. D. C., K. G. Murty, D. W. Sweeney, and C. Karel, "An Algorithm for the Traveling Salesman Problem," Operations Research, Vol. 11, 972-989 (1963).

27. Miller, C. E., A. W. Tucker, and R. A. Zemlin, "Integer Programming Formulation of Traveling Salesman Problems," T. Ass. Comput. March. 7, 326-329 (1960).

28. Parker, R. G., "Development of a Network Algorithm and its Application to Combinatorial Problems," Master's Thesis, Kansas State University, Manhattan, Kansas (1970).

29. Pierce, J. F., "Direct Search Algorithms for Truck-Dispatching Problems," Transportation Research, Vol. 3, No. 1, 1-42 (1969).

30. Raymond, T. C., "Heuristic Algorithm for the Traveling Salesman Problem," IBM J. Research and Development, 400-407 (July 1969).

31. Reiter, S., and G. Sherman, "Discrete Optimizing," SIAM 13, 864-889 (1965).

32. Sasieni M., A. Yaspan, and L. Friedman, Operations Research, John Wiley and Sons Inc., New York, 1964.

33. Shapiro, D., "Algorithms for the Solution of the Optimal Cost Traveling Salesman Problem," Ph.D. Dissertation Washington University, St. Louis, Mo. (1966).

34. Tillman, F. A., and H. Cochran, "A Heuristic Approach for Solving the Delivery Problem," Journal of Industrial Engineering, Vol. 19, No. 7, 354-358 (1968).

35. Tillman, F. A., and C. L. Hwang, "A Heuristic Approach for Solving the Multiterminal Delivery Problem," Unpublished Working Papers, Kansas State University, Manhattan, Kansas (1968).

36. Wagner, H. M., Principles of Operations Research, Prentice Hall Inc., Englewood Cliffs, New Jersey, 1969.

37. White, C., "An Algorithm for Finding Optimal or Near-Optimal Solutions to the Production Scheduling Problem," Ph.D. Dissertation, Purdue University, Lafayette, Indiana, 1963.

APPENDIX A

COMPUTER PROGRAM

This appendix includes a general discussion of the computer program devised for the traveling salesman problem, TSPCP. The algorithm developed in Chapter II is coded in this program. The TSPCP contains a main routine and six subroutines. The computer program write-up is also given. Details pertaining to the program itself as well as a program listing are furnished.

MAIN Routine

The MAIN routine serves as a control program. This routine reads all the information pertaining to the problem. After reading the distance matrix input cards, it prints out the complete original matrix. Assignments of links are all initialized to zero, such that

$$A(i,j) = 0, \qquad i, j = 1, 2, \ldots, n.$$

Control is then transferred to the MAXREG subroutine. Control is recovered at the end of the problem solving procedure. This whole process is repeated for every input problem.

REDUCE Subroutine

This subroutine performs the reduction of the distance matrix. Rows are reduced first such that for each non-deleted row i,

$$C(i,j) = C(i,j) - \min_{k=1,\ldots,n} [C(i,k)], \qquad j = 1, \ldots, n,$$

then, columns of the resulting matrix are reduced such that for each non-deleted column j,

$$C(i,j) = C(i,j) - \min_{k=1,\ldots,n} [C(k,j)], \qquad i = 1, \ldots, n.$$

This subroutine also finds the total amount of row and column reductions incurred during the current reduction process.

REGRET Subroutine

The regret subroutine evaluates the regrets corresponding to cells having zero value in the current reduced matrix, such that

$$R(I,J) = \min_{\substack{k \\ k \neq J}} [C(I,k)] + \min_{\substack{k \\ k \neq I}} [C(k,J)].$$

MAXREG Subroutine

The principal function of this subroutine is the assignment of links in the final route. A call statement to the CHOOSE subroutine serves to obtain the best candidate link, (I*, J*). This selected link is then tested to avoid closing any possible subtour by calling the BLOCK subroutine. Thus if the selected link is cleared for inclusion in the route, its assignment is made such that:

$$A(I^*, J^*) = 1,$$

then row I* and column J* are deleted, or simply

$$C(I^*, k) = C(k, J^*) = \infty, \qquad k = 1, \ldots, n,$$

link (J*, I*) is also deleted, such that

$$C(J^*, I^*) = \infty.$$

The value of the objective function is then

$$C = C + C(I^*, J^*)$$

where $C(I^*, J^*)$ is taken from the original distance matrix.

If level index L is such that

$$L \leq n-1,$$

its value is increased by 1 and the complete procedure is repeated. If $L = n$, the complete route has been constructed and the assigned links are ordered by calling the ROUTE subroutine. At the end of the problem, control is transferred back to the MAIN routine.

CHOOSE Subroutine

The function of this subroutine is to find the best candidate link to be tested for inclusion in the final route. It first attempts to find the link (I^*, J^*) with the largest regret such that:

$$R(I^*, J^*) = \max_{C(I,J) = 0} [R(I,J)].$$

If this selection fails to be unique, it applies the look ahead of finding the link (I^*, J^*) that gives rise to the smallest reduction of the matrix if included in the final route, or simply

$$D[I^*, J^*] = \min_{(I^*,J^*)} [D(I^*, J^*)].$$

where $D(I^*, J^*)$ is the reduction generated by link (I^*, J^*), if the tie still exists, the link with the largest J subscript is selected. After selection of the best candidate link (I^*, J^*) control is returned to the MAXREG subroutine.

BLOCK Subroutine

This subroutine tries to establish a sequence of links already assigned and the current candidate link. At level L < n-1, if this sequence forms a closed loop the link under consideration is deleted by making C(I*, J*) a prohibited link, and the original assignment A(I*, J*) = 0 remains unchanged, and control is returned to subroutine MAXREG; if the chain does not close a loop, the original assignment A(I*, J*) = 0 is changed to A(I*, J*) = 1, and control is returned to the MAXREG subroutine. This subroutine is not used at levels 1, 2, n-1, and n.

ROUTE Subroutine

This subroutine is called after n links have been assigned to the final route. It establishes the sequence of cities as they shall be visited by the salesman, under the convention that city 1 is the depot and terminal point of the route.

Computer Program Write-up

Purpose

The computer program presented in Appendix A has been designed to find one optimal or near optimal solution to the traveling salesman problem. The objective sought is to minimize the total distance traveled. This program can handle problems having symmetric as well as non-symmetric distance (or cost matrices)

Restrictions

This program has been coded in FORTRAN IV for the IBM 360/50 computer subject to the control of the WATFOR compiler. Present dimensions solve problems with up to 42 cities; for larger problems adjustments should be made.

Input Specification

The input data deck has two parts: (1) Control card, and (2) Data sets. The formal requirements of the input cards are listed below. The FORMAT statement for all input cards is FORMAT (16I5)

1. Control Card. This card establishes the number of problems to be solved. It represents the variable NPRB right-adjusted to column number 5 of the card.
2. Data sets. Each data set consists of two parts: (1) the problem information card, in which N, number of cities is right adjusted to column number 5 and ISYMM, the symmetry characteristic, is right adjusted to column 10., and (2) the distance matrix. If ISYMM = 0, the whole distance matrix should be given, taking care that the prohibited links, (I,I), I = 1, ..., N have the value 99999. Computer limitations do not allow us to introduce the value ∞ for these cells. If ISYM = 1, only the elements below the main diagonal of the distance matrix need to be given.

Output specification

The output is organized to provide: (1) information on the number of cities involved in the problem, (2) the original distance matrix, and (3) the route established for the given problem. The route constitutes the solution to the problem and shows the sequence of links assigned in the final route and the total distance traveled by the salesman.

EQUIVALENT NOTATIONS

Text	Computer Program
n	N
L	M
C(i,j)	NDIS(I,J)
A(i,j)	IT(I,J)
D	IREDUC
R(i,j)	IRGRET(I,J)

COMPUTER PROGRAM LISTING

```
C*****************************************************************
C
C      I.E. 896. RESEARCH IN INDUSTRIAL ENGINEERING.
C
C      HEURISTIC ALGORITHM FOR SOLVING THE TRAVELING SALESMAN
C      PROBLEM.
C
C*****************************************************************
C
C
C       PROGRAMMED BY
C                          JUAN F. VEGA
C                          DEPT. OF INDUSTRIAL ENGINEERING
C                          KANSAS STATE UNIVERSITY.
C
C
C*****************************************************************
C
C     EXPLANATION OF VARIABLES.
C
C          N           IS THE NUMBER OF CITIES.
C          ISYMM       EQUALS 1 IF THE DISTANCE MATRIX IS SYMME-
C                      TRIC AND ZERO OTHERWISE.
C          IT(I,J)     DECISION VARIABLE.IT HAS THE VALUE 1 WHEN
C                      LINK (I,J) IS ASSIGNED IN THE FINAL ROUTE
C                       AND ZERO OTHERWISE.
C          NPROB       IS THE NUMBER OF PROBLEMS TO BE SOLVED.
C          IDIST(I,J)ACTUAL COST FOR GOING FROM CITY I TO CITY
C                      J. IT CAN BE EXPRESSED IN UNITS OF TIME,
C                      DISTANCE,OR COST.
C          NDIS(I,J) ELEMENT OF AUXILIARY DISTANCE MATRIX UN -
C                      DERGOING TRANSFORMATIONS DURING THE PRO -
C                      BLEM SOLVING PROCEDURE.
C          M           LEVEL INDEX.
C          IREGRET(I,J)IS THE REGRET ASSOCIATED WITH LINK (I,J)
C          JCOL(J)     IF EQUALS 1 IT INDICATES THAT COLUMN J
C                      HAS BEEN DELETED FROM THE DISTANCE MATRIX
C                      OTHERWISE ITS VALUE IS ZERO.
C          IROW(I)     IF EQUALS 1 IT INDICATES THAT  ROW I HAS
C                      BEEN DELETED FROM THE DISTANCE MATRIX,O-
C                      THERWISE IT HAS THE VALUE 0.
C          IREDUC      IS THE AMOUNT OF REDUCTION UNDERGONE BY
C                      THE DISTANCE MATRIX.
C          IRMAX       IS THE MAXIMUM VALUE OF REGRET FOUND IN
C                      THE SEARCH FOR THE BEST CANDIDATE LINK.
C
C*****************************************************************
C
C      GENERAL INFORMATION.
C
```

```
C          ALL DATA CARDS WILL FOLLOW THE FORMAT(16I5).
C
C
C          PRESENT DIMENSIONS ALLOW THE SOLUTION OF PROBLEMS
C          WITH UP TO FORTY TWO CITIES.
C
C
C          INPUT OF DATA
C
C          1. THE FIRST DATA CARD WILL INDICATE THE NUMBER OF
C          PROBLEMS TO BE SOLVED.
C
C          2. FOR EACH PROBLEM THE FOLLOWING INFORMATION SHALL
C          BE GIVEN
C                    1. FIRST PROBLEM CARD INCLUDES NUMBER OF CI
C                    TIES AND IDENTIFICATION OF SYMMETRY.
C                    2. FOLLOWING CARDS WILL DENOTE THE DISTANCE
C                    MATRIX.FOR SYMMETRIC DISTANCE MATRICES IT
C                    IS ONLY REQUIRED TO PUNCH THE ELEMENTS BE-
C                    LOW THE MAIN DIAGONAL.
C
C*****************************************************************
C
C
C       MAIN PROGRAM
C
        COMMON IDIST(42,42),NDIS(42,42),IRECUC,IT(42,42)
        COMMON IROBET(42,42),IROW(42),JCOL(42)
C
  100   FORMAT(3X,I3,'  CITY TRAVELING SALESMAN PROBLEM')
  101   FORMAT(16I5)
  102   FORMAT(3X,' NONSYMMETRIC COST MATRIX',//)
  103   FORMAT(3X,' SYMMETRIC COST MATRIX',//)
  107   FORMAT(3X,'PROBLEM NUMBER ',I3)
  200   FORMAT(3X,'THE ORIGINAL DISTANCE MATRIX IS',/)
  202   FORMAT(3X,2016)
  203   FORMAT(3X,/)
        READ(1,101)NPROB
        DO 800 NPR=1,NPROB
        WRITE(3,107)NPR
        READ(1,101)N,ISYMM
        WRITE(3,100)N
        IF(ISYMM-1)104,105,104
  105   WRITE(3,103)
        DO 80 I=2,N
        IMINUS=I-1
        READ(1,101)(IDIST(I,J),J=1,IMINUS)
   80   CONTINUE
        DO 82 I=1,N
        IDIST(I,I)=99999
```

```
 82 CONTINUE
    DO 84 I=2,N
    IMINUS=I-1
    DO 84 J=1,IMINUS
    KAR=IDIST(I,J)

    IDIST(J,I)=KAR
 84 CONTINUE
    GO TO 106
104 WRITE(3,102)
    DO 108 I=1,N
    READ(1,101)(IDIST(I,J),J=1,N)
108 CONTINUE
106 DO 93 I=1,N
    IROW(I)=0
    JCOL(I)=0
    DO 93 J=1,N
    NDIS(I,J)=IDIST(I,J)
    IT(I,J)=0
 93 CONTINUE
    WRITE(3,200)
    DO 201 I=1,N
201 WRITE(3,202)(NDIS(I,J),J=1,N)
    WRITE(3,203)
    M=1
    CALL REDUCE(M,N)
    CALL REGRET(M,N)
    CALL MAXREG(M,N)
600 CONTINUE
    STOP
    END
```

```
      SUBROUTINE REDUCE(M,N)
C*****************************************************************
C
C         THIS SUBROUTINE REDUCES THE CURRENT DISTANCE MATRIX
C         IN ORDER TO GET,AT LEAST,ONE ZERO IN EVERY ROW AND
C         ONE ZERO IN EVERY COLUMN OF THE RESULTING MATRIX.
C
C*****************************************************************
C
      COMMON IDIST(42,42),NDIS(42,42),IREDUC,IT(42,42)
      COMMON IRGRET(42,42),IROW(42),JCOL(42)
C
C         REDUCE NON-DELETED ROWS.
C
      IREDUC=0
      DO 120 I=1,N
      MINROW=99999
      IF(IROW(I))160,160,120
  160 DO 130 J=1,N
  121 IF(NDIS(I,J)-MINROW)122,122,130
  122 MINROW=NDIS(I,J)
  130 CONTINUE
      IF(MINROW)120,120,119
  119 IREDUC=IREDUC+MINROW
      DO 140   J=1,N
      IF(NDIS(I,J)-99999)142,140,140
  142 NDIS(I,J)=NDIS(I,J)-MINROW
  140 CONTINUE
  120 CONTINUE
C
C         REDUCE NONDELETED COLUMNS.
C
      DO 220 J=1,N
      MINCOL=99999
      IF(JCOL(J))260,260,220
  260 DO 230 I=1,N
  221 IF(NDIS(I,J)-MINCOL)222,222,230
  222 MINCOL=NDIS(I,J)
  230 CONTINUE
      IF(MINCOL)220,220,219
  219 IREDUC=IREDUC+MINCOL
      DO 240   I=1,N
      IF(NDIS(I,J)-99999)242,240,240
  242 NDIS(I,J)=NDIS(I,J)-MINCOL
  240 CONTINUE
  220 CONTINUE
   22 RETURN
      END
```

```
      SUBROUTINE REGRET(M,N)
C*****************************************************************
C
C         THIS SUBROUTINE  FINDS THE REGRET FUNCTION CORRES-
C         PONDING TO CELLS WITH INTERCITY DISTANCE OF ZERO IN
C         THE CURRENT DISTANCE MATRIX.
C
C*****************************************************************
C
      COMMON IDIST(42,42),NDIS(42,42),IREDUC,IT(42,42)
      COMMON IRGRET(42,42),IROW(42),JCOL(42)
C
      DO 120 I=1,N
      DO 120 J=1,N
      IF(I-J)121,120,121
  121 IF(NDIS(I,J))120,110,120
  110 NDIS(I,J)=99999
      MINROW=99999
      MINCOL=99999
      DO 127 K=1,N
      IF(K-I)122,127,122
  122 IF(NDIS(K,J)-MINROW)126,127,127
  126 MINROW=NDIS(K,J)
  127 CONTINUE
      DO 128 K=1,N
      IF(K-J)124,128,124
  124 IF(NDIS(I,K)-MINCOL)129,128,128
  129 MINCOL=NDIS(I,K)
  128 CONTINUE
      NDIS(I,J)=0
      IRGRET(I,J)=MINROW+MINCOL
  120 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE MAXREG(M,N)
C*****************************************************************
C
C         THIS SUBROUTINE FINDS THE CELL WITH ZERO DISTANCE
C         HAVING THE LARGEST REGRET FUNCTION,ELIMINATES SUB-
C         TOURS AND ASSIGNS SELECTED LINKS INTO THE FINAL SO-
C         LUTION.
C
C*****************************************************************
C
      COMMON IDIST(42,42),NDIS(42,42),IREDUC,IT(42,42)
      COMMON IRGRET(42,42),IROW(42),JCOL(42)
C
      ITOTAL=0
      KAR=N-1
  100 CALL CHOOSE(M,N,IRMAX,IRIS,JRIS)
  160 IF(M-3)151,150,150
  151 IT(IRIS,JRIS)=1
      ITOTAL=ITOTAL+IDIST(IRIS,JRIS)
      DO 153 NONES=1,N
      NDIS(IRIS,NONES)=99999
      NDIS(NONES,JRIS)=99999
  153 CONTINUE
      NDIS(JRIS,IRIS)=99999
      M=M+1
      IROW(IRIS)=1
      JCOL(JRIS)=1
      CALL REDUCE (M,N)
      CALL REGRET(M,N)
      GO TO 100
  150 IF(M-KAR)158,158,159
  159 IT(IRIS,JRIS)=1
      ITOTAL=ITOTAL+IDIST(IRIS,JRIS)
      CALL ROUTE(N,ITOTAL)
      RETURN
  158 CALL BLOCK(M,N,IRIS,JRIS)
      IF(IT(IRIS,JRIS))171,171,152
  171 IRGRET(IRIS,JRIS)=-99999
      GO TO 100
  152 ITOTAL=ITOTAL+IDIST(IRIS,JRIS)
      DO 154 NONES=1,N
      NDIS(IRIS,NONES)=99999
      NDIS(NONES,JRIS)=99999
  154 CONTINUE
      NDIS(JRIS,IRIS)=99999
      IROW(IRIS)=1
      JCOL(JRIS)=1
      M=M+1
      CALL REDUCE (M,N)
      CALL REGRET(M,N)
```

```
GO TO 100
END
```

```
      SUBROUTINE CHOOSE(M,N,IRMAX,IRIS,JRIS)
C*************************************************************
C
C        THIS SUBROUTINE FINDS THE BEST CANDIDATE LINK TO BE
C        TESTED FOR INCLUSION IN THE FINAL ROUTE.
C
C*************************************************************
C
      COMMON IDIST(42,42),NDIS(42,42),IRECUC,IT(42,42)
      COMMON IRGRET(42,42),IROW(42),JCOL(42)
      DIMENSION NN(42,42)
C
      INDEX=1
  202 IRMAX=0
      DO 120 I=1,N
      DO 120 J=1,N
      IF(I-J)121,120,121
  121 IF(NDIS(I,J))120,122,120
  122 IF(IRGRET(I,J))120,115,115
  115 IF(IRGRET(I,J)-IRMAX)120,111,111
  111 IRMAX=IRGRET(I,J)
      IRIS=I
      JRIS=J
      INDEX=INDEX+1
  120 CONTINUE
      IF(INDEX-1)321,321,131
  131 KAR=N-1
      IF(M-KAR)310,331,321
  331 DO 332 I=1,N
      DO 332 J=1,N
      IF(NDIS(I,J))332,333,332
  333 IF(IRGRET(I,J)-IRMAX)332,334,332
  334 IRIS=I
      JRIS=J
      GO TO 321
  332 CONTINUE
  310 ISAVE=99999
      DO 301 I=1,N
      DO 301 J=1,N
      NN(I,J)=NDIS(I,J)
  301 CONTINUE
      DO 320 I=1,N
      DO 320 J=1,N
      IF(NDIS(I,J))320,322,320
  322 IF(IRGRET(I,J)-IRMAX)320,323,320
  323 IROW(I)=1
      JCOL(J)=1
      NDIS(J,I)=99999
      DO 400 K=1,N
      NDIS(I,K)=99999
```

```
400 NPIS(K,J)=99999
    CALL REDUCE(M,N)
    IF(IREDUC.GE.ISAVE)GO TO 330
    IRIS=I
    JRIS=J
330 IROW(I)=0
    JCOL(J)=0
    DO 315 IK=1,N
    DO 315 KK=1,N
    NPIS(IK,KK)=NN(IK,KK)
315 CONTINUE
    ISAVE=IREDUC
320 CONTINUE
321 RETURN
    END
```

```
      SUBROUTINE BLOCK(M,N,IRIS,JRIS)
C*****************************************************************
C
C         THIS SUBROUTINE ELIMINATES THE OCCURRENCE OF TOURS
C         NOT INCLUDING ALL THE CITIES BEFORE COMPLETION OF
C         THE COMPUTATIONAL PROCEDURE.
C
C*****************************************************************
C
      COMMON IDIST(42,42),NDIS(42,42),IREDUC,IT(42,42)
      COMMON IRGRET(42,42),IROW(42),JCOL(42)
C
      ILO=M-1
   21 LEAD=JRIS
      ICOUNT=1
   72 IF(ICOUNT.GT.ILO)GO TO 23
      DO 122 KE=1,N
      IF(IT(LEAD,KE)-1)122,123,122
  123 LEAD=KE
      IF(LEAD.EQ.IRIS)GO TO 74
      ICOUNT=ICOUNT+1
      GO TO 72
  122 CONTINUE
      GO TO 23
   74 IRGRET(IRIS,JRIS)=-99999
      NDIS(IRIS,JRIS)=99999
      RETURN
   23 IT(IRIS,JRIS)=1
      RETURN
      END
```

```
      SUBROUTINE RCUTE(N,ITOTAL)
C*****************************************************************
C
C         THIS SUBROUTINE ESTABLISHES A SEQUENCING OF LINKS
C         BETWEEN CITIES SO AS TO BUILD A TOUR STARTING AND
C         FINISHING AT THE DEPOT.
C
C*****************************************************************
C
      COMMON IDIST(42,42),NDIS(42,42),IREDUC,IT(42,42)
      COMMON IRGRET(42,42),IROW(42),JCOL(42)
      DIMENSION IPOINT(42)
C
  100 FORMAT(//,3X,'THE FOLLOWING ROUTE IS ESTABLISHED',//)
  101 FORMAT(3X,'LINK NUMBER',5X,'FROM CITY',5X,'TO',3X,'CITY')
  102 FORMAT(3X,42('*'))
  103 FORMAT(3X,'*',4X,I3,11X,I3,11X,I3,5X,'*')
  104 FORMAT(//,3X,'TOTAL DISTANCE TRAVELED IS',2X,I6)
  105 FORMAT('1')
      NMINUS=N-1
      K=2
      IPOINT(1)=1
      DO 120 J=1,N
      IF(IT(1,J)-1)120,121,120
  121 LEAD=J
      IPOINT(2)=J
      GO TO 125
  120 CONTINUE
  125 K=K+1
      IF(K.GT.N) GO TO 124
      DO 122 KE=1,N
      IF(IT(LEAD,KE)-1)122,123,122
  123 LEAD=KE
      IPOINT(K)=KE
      GO TO 125
  122 CONTINUE
  124 WRITE(3,100)
      WRITE(3,101)
      WRITE(3,102)
      DO 130 I=1,NMINUS
      IPLUS=I+1
  130 WRITE(3,103)I,IPOINT(I),IPOINT(IPLUS)
      WRITE(3,103)N,IPOINT(N),IPOINT(1)
      WRITE(3,102)
      WRITE(3,104)ITOTAL
      WRITE(3,105)
      RETURN
      END
```

APPENDIX B

PROBLEMS

This Appendix presents the relevant information concerning the problems solved by application of the computer program prepared for the proposed heuristic algorithm.

PROBLEM 1

1. Size: n = 4.

2. Type: Non-symmetrical.

3. Distance matrix:

FROM \ TO	1	2	3	4
1	-	4	2	3
2	5	-	2	6
3	3	5	-	4
4	4	3	5	-

4. Optimal route: $1 \rightarrow 4 \rightarrow 2 \rightarrow 3 \rightarrow 1$

   Total distance: 11

5. Route found by the heuristic algorithm: $1 \rightarrow 4 \rightarrow 2 \rightarrow 3 \rightarrow 1$

   Total distance: 11

6. Quality of solution: 1.00

7. Computer time on the IBM 360/50: 2.01 seconds

PROBLEM 2.

1. Size: n = 5.

2. Type: Non-symmetric

3. Distance matrix:

FROM \ TO	1	2	3	4	5
1	∞	5	13	13	15
2	2	∞	3	5	2
3	7	5	∞	1	3
4	5	9	4	∞	7
5	5	2	8	4	∞

4. Optimal route: 1 → 2 → 3 → 5 → 4 → 1

   Total distance: 20

5. Route found by the heuristic algorithm: 1 → 2 → 5 → 3 → 4 → 1

   Total distance: 21

6. Quality of solution: 0.95

7. Computer time on the IBM 360/50: 3.09 seconds

PROBLEM 3

1. Size: n = 5.

2. Type: Nonsymmetric

3. Distance matrix:

FROM \ TO	1	2	3	4	5
1	-	2	5	7	1
2	6	-	3	8	2
3	8	7	-	4	7
4	12	4	6	-	5
5	1	3	2	8	-

4. Optimal route: 1 → 2 → 3 → 4 → 5 → 1

   Total distance: 15

5. Route found by the heuristic algorithm: 1 → 2 → 3 → 4 → 5 → 1.

   Total distance: 15

6. Quality of solution: 1.00

7. Computer time on the IBM 360/50: 3.30 seconds

PROBLEM 4

1. Size: n = 5

2. Type: Nonsymmetric

3. Distance matrix:

FROM \ TO	1	2	3	4	5
1	-	10	25	25	10
2	1	-	10	15	2
3	8	9	-	20	10
4	14	10	24	-	15
5	10	8	25	27	-

4. Optimal route: 1 → 5 → 2 → 3 → 4 → 1

   Total distance: 62

5. Route found by the heuristic algorithm: 1 → 5 → 4 → 2 → 3 → 1

   Total distance: 65

6. Quality of solution: 0.95

7. Computer time on the IBM 360/50: 2.62

PROBLEM 5

1. Size: n = 6.

2. Type: Non-symmetric.

3. Distance matrix:

FROM \ TO	1	2	3	4	5	6
1	-	27	43	16	30	26
2	7	-	16	1	30	25
3	20	13	-	35	5	0
4	21	16	25	-	18	18
5	12	46	27	48	-	5
6	23	5	5	9	5	-

4. Optimal route: 1 → 4 → 3 → 5 → 6 → 2 → 1

   Total distnace 63.

5. Route found by the heuristic algorithm: 1 → 4 → 3 → 5 → 6 → 2 → 1

   Total distance: 63

6. Quality of solution: 1.00.

7. Computer time on the IBM 360/50: 5.28 seconds.

PROBLEM 6

1. Size: n = 7.

2. Type: Symmetric

3. Distance matrix:

FROM \ TO	1	2	3	4	5	6	7
1	-	30	40	55	28	17	33
2	30	-	29	68	35	20	56
3	40	29	-	47	22	24	46
4	55	68	47	-	31	51	27
5	28	35	22	31	-	20	25
6	17	20	24	51	20	-	36
7	33	56	46	27	25	36	-

4. Optimal route: 1 → 6 → 2 → 3 → 5 → 4 → 7 → 1

   Total distance: 179

5. Route found by the heuristic algorithm: 1 → 6 → 2 → 3 → 5 → 4 → 7 → 1

   Total distance: 179

6. Quality of solution: 1.00

7. Computer time on the IBM 360/50: 6.81 seconds

PROBLEM 7

1. Size: n = 10.

2. Type: Non-symmetric.

3. Distance Matrix:

FROM \ TO	1	2	3	4	5	6	7	8	9	10
1	-	51	55	90	41	63	77	69	0	23
2	50	-	0	64	8	53	0	46	73	72
3	30	77	-	21	25	51	47	16	0	60
4	65	0	6	-	2	9	17	5	26	42
5	0	94	0	5	-	0	41	31	59	48
6	79	65	0	0	15	-	17	47	32	43
8	0	17	9	27	46	15	84	-	0	24
9	56	7	45	39	0	93	67	79	-	38
10	30	0	42	56	49	77	76	49	23	-

4. Optimal route: 1 → 9 → 5 → 6 → 4 → 7 → 10 → 2 → 3 → 8 → 1

   Total distance: 33.

5. Route found by the heuristic algorithm: 1 → 9 → 5 → 6 → 4 → 7 → 10 → 2 → 3 → 8 → 1

   Total distance: 33

6. Quality of solution: 1.00

7. Computer time on the IBM 360/50: 17.96 seconds

PROBLEM 8

1. Size: n = 6

2. Type: Symmetric

3. Distance matrix:

FROM \ TO	1	2	3	4	5	6
1	-	11	9	12	13	10
2	11	-	10	11	4	8
3	9	10	-	8	9	4
4	12	11	8	-	7	2
5	13	4	9	7	-	5
6	0	8	4	2	5	-

4. Best known route: 1 → 3 → 6 → 4 → 5 → 2 → 1

   Total distance 37

5. Route found by the heuristic algorithm: 1 → 2 → 5 → 4 → 6 → 3 → 1

   Total distance 37

6. Quality of solution 1.00

7. Computer time on the IBM 360/50: 4.80

PROBLEM 9

1. Size: n = 5.

2. Type: Symmetric.

3. Distance matrix:

FROM \ TO	1	2	3	4	5
1	-	30	26	50	40
2	30	-	24	40	50
3	26	24	-	24	26
4	50	40	24	-	30
5	40	50	26	30	-

4. Optimal route: $1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 1$

   Total distance: 148

5. Route pound by the heuristic algorithm: $1 \rightarrow 3 \rightarrow 5 \rightarrow 4 \rightarrow 2 \rightarrow 1$.

   Total distance: 152.

6. Quality of solution:

7. Computer time on the IBM 360/50: 5.20 seconds.

PROBLEM 10

1. Size: n = 5.

2. Type: Symmetric.

3. Distance matrix:

FROM \ TO	1	2	3	5	5
1	-	10	8	4	2
2	10	-	11	9	12
3	8	11	-	10	11
4	4	9	10	-	8
5	2	12	11	8	-

4. Best known route: 1 → 5 → 4 → 2 → 3 → 1

   Total distance: 38

5. Route found by the heuristic algorithm: 1 → 4 → 2 → 3 → 5 → 1

   Total distance 37

6. Quality of solution: 1.00

7. Computer time on the IBM 360/50: 2.87 seconds

PROBLEM 11

1. Size: n = 10.

2. Type: Symmetric

3. Distance matrix:

FROM \ TO	1	2	3	4	5	6	7	8	9	10
1	-	28	57	72	81	85	80	113	89	80
2	28	-	28	45	54	57	63	85	63	63
3	57	28	-	20	30	28	57	57	40	57
4	72	45	20	-	10	20	72	45	20	45
5	81	54	30	10	-	22	81	41	10	41
6	85	57	28	20	22	-	63	28	28	63
7	80	63	57	72	81	63	-	30	89	113
8	113	85	57	45	41	28	80	-	40	80
9	89	63	40	20	10	28	89	40	-	40
10	80	63	57	45	41	63	113	80	40	-

4. Optimal route: 1 → 2 → 3 → 4 → 5 → 10 → 9 → 8 → 6 → 7 → 1

   Total distance: 378

5. Route found by the heuristic algorithm: 1 → 2 → 3 → 4 → 5 → 10 → 9 → 8 → 6 → 7 → 1

   Total distance: 378

6. Quality of solution 1.00

7. Computer time on the IBM 360/50: 14.62 seconds

PROBLEM 12

1. Size n = 6

2. Type: Nonsymmetric

3. Distance matrix:

FROM \ TO	1	2	3	4	5	6
1	-	1	7	3	14	2
2	3	-	6	9	1	24
3	6	14	-	3	7	3
4	2	3	5	-	9	11
5	15	7	11	2	-	4
6	20	5	13	4	18	-

4. Optimal route: 1 → 3 → 6 → 2 → 5 → 4 → 1

   Total distance 20

5. Route found by the heuristic algorithm 1 → 3 → 6 → 2 → 5 → 4 → 1

   Total distance 20

6. Quality of solution: 1.00

7. Computer time on the IBM 360/50: 4.00

PROBLEM 13

1. Size: n = 5

2. Type: Symmetric

3. Distance matrix:

FROM \ TO	1	2	3	4	5
1	-	702	814	294	2128
2	702	-	876	554	2229
3	814	876	-	249	2833
4	294	554	249	-	1884
5	2128	2229	2833	1884	-

4. Optimal route: 1 → 4 → 5 → 2 → 3 → 1

   Total distance 6097

5. Route found by the heuristic algorithm: 1 → 3 → 2 → 5 → 4 → 1

   Total distance: 6097

6. Quality of solution 1.00

7. Computer time on the IBM 360/50: 2.46 seconds.

PROBLEM 14

1. Size: n = 42

2. Type: Symmetric

3. Distance matrix: (shown on next page)

4. Optimal route: 1 2 3 ... 41 42 1.

   Total distance: 699

5. Route found by the heuristic algorithm: 1 → 2 → 6 → 7 → 9 → 8 → 17 → 13 → 14 → 15 → 16
   18 →19 →20 →21 →22 →23 → 12 → 11 → 10 → 25 → 24
   27 →26 →31 →30 →28 →29 → 33 → 32 → 34 → 35 →
   36 →37 →38 →39 →40 → 5 → 4 → 3 → 41 → 42 → 1.

   Total distance: 802

6. Quality of solution: .85

7. Computer time on the IBM 360/50: 273.35

Distance Matrix for Problem 1½ (taken directly from [12]).

TABLE I

ROAD DISTANCES BETWEEN CITIES IN ADJUSTED UNITS

The figures in the table are mileages between the two specified numbered cities, less 11, divided by 17, and rounded to the nearest integer.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41
2	8																																								
3	39	45																																							
4	37	47	9																																						
5	50	49	21	15																																					
6	61	62	21	20	17																																				
7	58	60	16	17	18	6																																			
8	59	63	15	20	26	17	10																																		
9	62	66	20	25	31	22	15	5																																	
10	81	81	40	44	50	41	35	24	20																																
11	103	107	62	67	72	63	57	46	41	23																															
12	108	117	66	71	77	68	61	51	46	26	11																														
13	145	149	104	108	114	106	99	88	84	63	49	40																													
14	181	185	140	144	150	142	135	124	120	99	85	76	35																												
15	187	191	146	150	156	142	137	130	125	105	90	81	41	10																											
16	161	170	120	124	130	115	110	104	105	90	72	64	34	31	27																										
17	142	146	101	104	111	97	91	85	86	75	51	59	29	53	48	21																									
18	174	178	133	138	143	129	123	117	118	107	83	84	54	46	35	26	31																								
19	185	186	142	143	140	130	126	124	128	118	93	101	72	69	58	58	43	26																							
20	164	165	120	123	124	106	106	105	110	104	86	97	71	93	82	62	42	45	22																						
21	137	139	94	96	94	80	78	77	84	77	56	64	65	90	87	58	36	68	50	30																					
22	117	122	77	80	83	68	62	60	61	50	34	42	49	82	77	60	30	62	70	49	21																				
23	114	118	73	78	84	69	63	57	59	48	28	36	43	77	72	45	27	59	69	55	27	5																			
24	85	89	44	48	53	41	34	28	29	22	23	35	69	105	102	74	56	88	99	81	54	32	29																		
25	77	80	36	40	46	34	27	19	21	14	29	40	77	114	111	84	64	96	107	87	60	40	37	8																	
26	87	89	44	46	46	30	28	29	32	27	36	47	78	116	112	84	66	98	95	75	47	36	39	12	11																
27	91	93	48	50	48	34	32	33	36	30	34	45	77	115	110	83	63	97	91	72	44	32	36	9	15	3															
28	105	106	62	63	64	47	46	49	54	48	46	59	85	119	115	88	66	98	79	59	31	36	42	28	33	21	20														
29	111	113	69	71	66	51	53	56	61	57	59	71	96	130	126	98	75	98	85	62	38	47	53	39	42	29	30	12													
30	91	92	50	51	46	30	34	38	43	49	60	71	103	141	136	109	90	115	99	81	53	61	62	36	34	24	28	20	20												
31	83	85	42	43	38	22	26	32	36	51	63	75	106	142	140	112	93	126	108	88	60	64	66	39	36	27	31	28	28	8											
32	89	91	55	55	50	34	39	44	49	63	76	87	120	155	150	123	100	123	109	86	62	71	78	52	49	39	44	35	24	15	12										
33	95	97	64	63	56	42	49	56	60	75	86	97	126	160	155	128	104	128	113	90	67	76	82	62	59	49	53	40	29	25	23	11									
34	74	81	44	43	35	23	30	39	44	62	78	89	121	159	155	127	103	136	124	101	75	79	81	54	50	42	46	43	39	23	14	14	21								
35	67	69	42	41	31	25	32	41	46	64	83	90	130	164	160	133	114	146	134	111	85	84	86	59	52	47	51	53	49	32	24	24	30	9							
36	74	76	61	60	42	44	51	60	66	83	102	110	147	185	179	155	133	159	146	122	98	105	107	79	71	66	70	70	60	48	40	36	33	25	18						
37	57	59	46	41	25	30	36	47	52	71	93	98	136	172	172	148	126	158	147	124	121	97	99	71	65	59	63	67	62	46	38	37	43	23	13	17					
38	45	46	41	34	20	34	38	48	53	73	96	99	137	176	178	151	131	163	159	135	108	102	103	73	67	64	69	75	72	54	46	49	54	34	24	29	12				
39	35	37	35	26	18	34	36	46	51	70	93	97	134	171	176	151	129	161	163	139	118	102	101	71	65	65	70	84	78	58	50	56	62	41	32	38	21	9			
40	29	33	39	21	18	35	33	40	45	65	87	91	117	166	171	144	125	157	156	139	113	95	97	67	60	62	67	79	82	62	53	59	66	45	38	45	27	15	6		
41	3	11	41	37	47	57	55	58	63	83	105	109	147	186	188	164	144	176	182	161	134	119	116	86	78	84	88	101	108	88	80	86	92	71	64	71	54	41	32	35	
42	5	12	55	41	53	64	61	61	66	84	111	113	150	186	192	166	147	180	183	167	140	124	119	90	87	90	94	107	114	77	86	92	98	80	74	77	60	48	38	32	6

A HEURISTIC ALGORITHM FOR TRAVELING SALESMAN PROBLEM

by

JUAN FELICIANO VEGA VILLALOBOS

B.S., National University of Engineering, Peru, 1961

AN ABSTRACT OF A MASTER'S THESIS

submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

Department of Industrial Engineering

KANSAS STATE UNIVERSITY

Manhattan, Kansas

1970

The traveling salesman problem consists of finding the minimal distance route starting and ending at the same terminal and visiting, once and only once, each of a number of cities.

The main objective of this research was to investigate some procedures leading to the development of optimal or near-optimal solutions in order to reduce the computational effort and storage requirements.

A heuristic algorithm was developed for the solution of the traveling salesman problem using an approach similar to that of the branch-and-bound without backtracking. Assignment of city-pairs in the route is made according to heuristic rules involving the use of regrets and the application of a look ahead rule to break the ties among candidate links. Several problems were solved using the computer program which was coded in FORTRAN IV for the IBM 360/50. Detailed solution by the proposed algorithm is given for a 10-city sample problem, for which an optimal solution was obtained.

Several ramifications of the algorithm were attempted for the traveling salesman problem. It was established that there is no similarity between the savings and regrets as decision criteria for the assignment of priorities in the selection of candidate links.

A HEURISTIC ALGORITHM FOR TRAVELING SALESMAN PROBLEMS

by

JUAN FELICIANO VEGA

B.S., National University of Engineering, Peru, 1961

AN ABSTRACT OF A MASTER'S THESIS

submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

Department of Industrial Engineering

KANSAS STATE UNIVERSITY

Manhattan, Kansas

1970