THE TRAVELING SALESMAN PROBLEM: A SURVEY†

M. Bellmore and G. L. Nemhauser

The Johns Hopkins University, Baltimore, Maryland

(Received June 24, 1966)

A survey and synthesis of research on the traveling salesman problem is given. We begin by defining the problem and presenting several theorems. This is followed by a general classification of the solution techniques and a detailed description of some of the proven methods. Finally a summary of computational results is given.

IN THE traveling salesman problem we are given a nonnegative integer n and an n-dimensional square matrix $C=\{c_{ij}\}$. Any sequence of $p+1$ integers taken from $(1, 2, \cdots, n)$, in which each of the n integers appears at least once and the first and last integers are identical is called a tour. A tour may be written as‡

$$t=(i_1, i_2, i_3, \cdots, i_{p-1}, i_p, i_1).$$

By a feasible solution to the traveling salesman problem, we mean a tour. An optimal solution is a tour such that

$$z(t)=\sum_{(i,j)\epsilon t'} c_{ij} \text{ is minimized,}$$

where $$t'=[(i_1, i_2), (i_2, i_3), \cdots, (i_{p-1}, i_p), (i_p, i_1)]$$

is the ordered pair representation of t.

The usual terminology is that the n integers correspond to cities or nodes, the ordered pairs (i,j) are links or arcs joining the nodes, and c_{ij} is the 'distance' from node i to node j, or the length of arc (i,j). The tour t is a closed path passing through each node at least once. The length of the tour, denoted by $z(t)$, is the sum of the arc lengths over the arcs included in the tour.

DISTANCE MEASURES

A SUBTOUR s,

$$s=(i_1, i_2, i_3, \cdots, i_k, i_1)$$

† Partially supported by National Bureau of Standards under Contract CST-348 to The Johns Hopkins University.

‡ Occasionally we will write $t=(i_1, i_2, \cdots, i_{p-1}, i_p)$ with the return to i_1 implied.

is a closed path that does *not* pass through *all* of the n nodes [$(i_1, i_2, \cdots, i_k)$ are distinct and $k<n$]. To have a meaningful problem the length of every subtour must be nonnegative, i.e.,

$$z(s) = \sum_{(i,j)\epsilon s'} c_{ij} \geqq 0,$$

where s' is the ordered pair representation of s. In fact, if any subtour had a negative length, $z(t)$ could be made arbitrarily small by having s appear an infinite number of times in t. We shall assume that the length of every subtour is nonnegative (note this implies that $c_{ii} \geqq 0$). Clearly this will be the case if $c_{ij} \geqq 0$ for all i and j.

In general, except for the nonnegative length of every subtour, the elements c_{ij} are completely arbitrary. However, there are some restrictions on the distance measures that admit important theoretical results and/or superior computational procedures. They are symmetry, triangle inequality, Euclidean plane and a particular 'distance' function suitable for certain job sequencing problems proposed by GILMORE AND GOMORY[15] (*see* the subsection, "Job Sequencing and the Gilmore-Gomory Algorithm.")

THEORETICAL RESULTS

PRESENTLY, there is not an adequate theory for the traveling salesman problem. We mean this in the sense that solutions to the traveling salesman problem cannot, in general, be found as efficiently as they can, for example, for a shortest route problem of comparable size. Nevertheless, there is some theory for the traveling salesman problem that, in part, has made it possible to construct the algorithms that presently exist.

In this section, we state, without proof, and interpret the main theorems. Most of them are rather obvious.

The first four theorems yield exploitable properties of the distance measure.

Theorem 1[18]

If C satisfies the triangle inequality, there is an optimal tour in which each node is visited once and only once.

Note that the triangle inequality is always satisfied if we replace c_{ij} by c'_{ij} where c'_{ij} is the length of a shortest path from i to j. Thus, a problem with an arbitrary distance matrix C can be solved assuming that each city is visited exactly once by replacing C by C'. If (i_p, i_q) is an arc in the optimal tour under C' and $(i_p, i_s, i_t, \cdots, i_q)$ is a shortest route from i_p to i_q under C, then there is an optimal tour under C that contains the sequence $(i_p, i_s, i_t, \cdots, i_q)$.

This result is important because almost all algorithms for the traveling salesman problem are designed to find the minimal length tour that goes through each node exactly once. Hereafter we shall assume that C is chosen so that there exists an optimal tour in which each node is visited exactly once or that the problem is to find an optimal tour under this restriction.

When C satisfies the Euclidean distance measure in two dimensions, Theorem 1 can be strengthened. Let the nodes be represented by points in two-dimensional space in such a way that the distance between nodes i and j is c_{ij}.

Theorem 2[3,14]

There exists an optimal tour that does not cross itself when C satisfies the Euclidean distance measure.

Theorem 3†[3]

Let G be the convex hull of the points in two-dimensional Euclidean space. There exists an optimal tour in which the relative order of the points on the boundary of G is preserved.

Theorems 2 and 3 serve to exclude from consideration those tours that do not satisfy these properties. Several algorithms can be improved by taking advantage of these properties.

For symmetric, but not necessarily Euclidean problems, we can exclude half of the tours since:

Theorem 4

If C is symmetric and t_1 and t_2 are two tours in which the nodes are visited in reverse order, that is

$$t_1=(i_1, i_2, \cdots, i_n, i_1), \qquad t_2=(i_1, i_n, \cdots, i_2, i_1)$$

then $z(t_1)=z(t_2)$.

In general there are $(n-1)!$ tours, but for symmetric problems only $(n-1)!/2$ need to be considered.

The next few theorems are useful in constructing bounds on the length of an optimal tour. The 'assignment problem,' which is relatively easy to solve may be stated as

$$\min w=\sum_{i=1}^{i=n}\sum_{j=1}^{j=n} c_{ij}x_{ij},$$

subject to:

$$\sum_i x_{ij}=1, \qquad (j=1, \cdots, n)$$

$$\sum_j x_{ij}=1, \qquad (i=1, \cdots, n)$$

$$x_{ij}=0, 1. \qquad (\text{all } i \text{ and } j)$$

† Actually this theorem is a slight generalization of BARACHET's Theorem 3.

Theorem 5

Let t be any tour in which each node is visited exactly once and $x_{ij}=0$ if (i, j) is not in the tour and $x_{ij}=1$ if (i, j) is in the tour. Then $\{x_{ij}\}$ is a feasible solution to the assignment problem with

$$z(t)=\sum_{i=1}^{i=n}\sum_{j=1}^{j=n}c_{ij}x_{ij}=w.$$

Unfortunately the converse is not true; feasible solutions to the assignment problem may not be tours. For example, with $n=4$, $x_{12}=x_{21}=x_{34}=x_{43}=1$ is a feasible solution to the assignment problem, but when interpreted for the traveling salesman problem, yields the subtours (1, 2, 1) and (3, 4, 3). As a result of Theorem 5 we have that min $w\leqq z(t)$, over all tours.

Theorem 6

Let k_p and k_q be real numbers associated with a fixed pair of nodes p and q such that:

$$\begin{aligned}
c'_{pj}&=c_{pj}-k_p, && (j=1, \cdots, n;\ j\neq q)\\
c'_{iq}&=c_{iq}-k_q, && (i=1, \cdots, n;\ i\neq p)\\
c'_{pq}&=c_{pq}-k_p-k_q,\\
c'_{ij}&=c_{ij}, \text{ otherwise,}
\end{aligned}$$

and $z'(t)$ be the length of tour t under C'. Then $z'(t)=z(t)-k_p-k_q$.

Theorem 6 allows us conveniently to work with reduced matrices with as many zero elements as possible. Closely related to Theorem 6 is Theorem 7.

Theorem 7

If tour t does not contain arc (p, q) $z(t)\geqq h_p+h_q$ where $h_p=\min_{j\neq p,q} c_{pj}$ and $h_q=\min_{i\neq p,q} c_{iq}$.

Theorem 8

Let p_1 and p_2 be two different permutations of the integers $(2, 3, \cdots, k+1)$;

$$p_1=(i_1, i_2, \cdots, i_k), \qquad p_2=(j_1, j_2, \cdots, j_k)$$

and let $$z(p_m)=\sum_{(i,j)\in p_m'} c_{ij}, \qquad (m=1, 2)$$

where p_m' is the ordered pair representation of p_m. Then, if

$$c_{1i_1}+z(p_1)+c_{i_k s}<c_{1j_1}+z(p_2)+c_{j_k s},$$

then $(1, p_2, s)$ cannot be a segment of an optimal tour.

By applying Theorem 8 recursively we can develop functional equations for determining an optimal tour.

We have noted that all tours are feasible solutions to the assignment problem, but that, in addition to tours, subtours are feasible solutions to the assignment problem. Since the assignment problem is a linear program in 0-1 variables, if we can find linear constraints in the variables $\{x_{ij}\}$ that will exclude all subtours but no tours, then the traveling salesman problem can be written as a linear program in 0–1 variables. Theorems 9 and 10 accomplish this.

Theorem 9[9,10,28]

Let S, $\bar{S}$ be a partition of the integers $i=1, \cdots, n$: i.e., $S \cap \bar{S} = \emptyset$ and $S \cup \bar{S} = \{1, 2, \cdots, n\}$. For symmetric distances let $x_{ij}=0$ if undirected arc (i, j) is not in a tour and $x_{ij}=1$ if undirected arc (i, j) is in a tour. An optimal tour can be found by solving the integer program

$$\min z = \sum_{j=2}^{j=n} \sum_{i=1}^{j-1} c_{ij} x_{ij},$$

subject to: $x_{ij}=0, 1, \quad (i=1, \cdots, j-1; j=2, \cdots, n)$

and the loop constraints

$$\sum_{i \in S} \sum_{j \in \bar{S}} x_{ij} \geqq 2,$$

for all nonempty partitions $(S, \bar{S})$ such that if $(S, \bar{S})$ is considered $(\bar{S}, S)$ is not.

The loop constraints are illustrated in Fig. 1; (a) contains a subtour, (b) and (c) do not. There are $2^{n-1}-1$ of these constraints in an n-city problem. Assymmetric problems require twice as many variables and loop constraints. Specifically, the loop constraints are

$$\sum_{i \in S} \sum_{j \in \bar{S}} x_{ij} \geqq 1,$$

for all nonempty partitions $(S, \bar{S})$, where $x_{ij}=1$, (0) if directed arc (i, j) is in (not in) a tour.

A different formulation that reduces the traveling salesman problem to an integer program is given in Theorem 10.

Theorem 10[29]

Let $x_{ij}=1$, (0) if directed arc (i, j) is in (not in) a tour. An optimal tour can be found by solving the integer program

$$\min z = \sum_{i=1}^{i=n} \sum_{j=1}^{j=n} c_{ij} x_{ij},$$

$$\sum_{i=1}^{i=n} x_{ij} = 1, \qquad \text{all } j \text{ except } j=i_0 \ (i_0 \text{ arbitrary}),$$

$$\sum_{j=1}^{j=n} x_{ij} = 1, \qquad \text{all } i \text{ except } i=i_0,$$

$$u_i - u_j + n x_{ij} \leqq n-1, \qquad \text{all } i \text{ and } j \text{ except } i=j=i_0.$$

This formulation requires approximately n^2 constraints, considerably fewer than the formulation of Theorem 9. However, this does not necessarily mean that the formulation of Theorem 10 is easier to solve. We shall discuss this point further in the section on algorithms.

Given a set of nodes, $i=1, \cdots, n$ where c_{ij} is the length of the arc joining nodes i and j, the longest path problem is to find a simple path between two distinguished nodes, say 1 and n, such that the sum of the arc lengths is maximum. Theorem 11 allows one to solve the traveling salesman problem as a longest path problem.

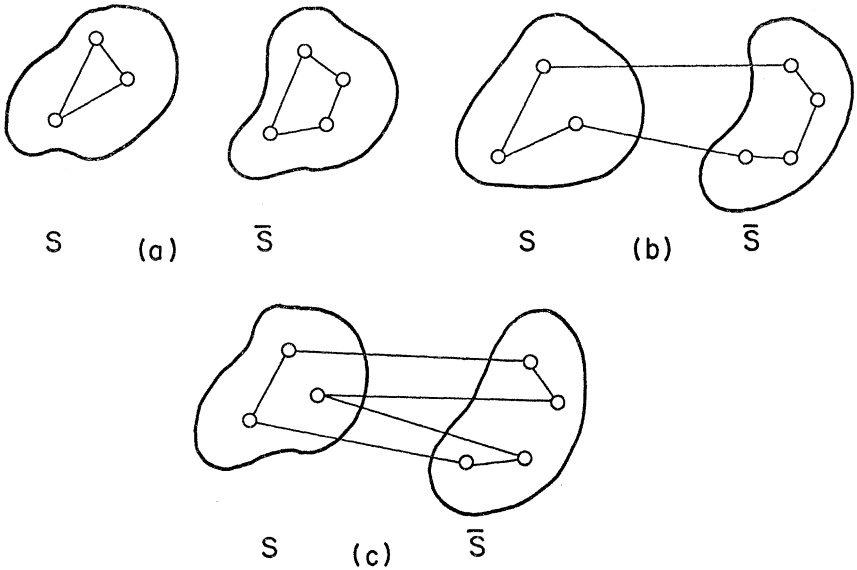

Fig. 1. Elimination of subtours.

Theorem 11[18]

Given the nodes ($i=1, \cdots, n$), arcs (i, j) and distance matrix C construct a new network containing the nodes and arcs from the original network plus one additional node, denoted by α, and an additional arc (j, α) for each j such that $(i, 1)$ is an arc in the original network. The distances d_{ij} in the new network are:

$$d_{ii}=0, \text{ for all } i,$$

$$d_{j1}=-\infty, \text{ for all } j\neq 1,$$

$$d_{j\alpha}=k-c_{j1}, \text{ for all } j\neq\alpha,$$

$$d_{ij}=k-c_{ij}, \text{ otherwise,}$$

where k is any finite number $>$ sum of n largest c_{ij}.

A longest path from 1 to α in the new network contains every intermediate node $(2, \cdots, n)$ and if $(1, i_1, \cdots, i_{n-1}, \alpha)$ is such a longest path $(1, i_1, \cdots, i_{n-1}, 1)$ is an optimal tour.

Theorem 11 has not proved to be useful since no efficient algorithms for the longest path problem have been discovered yet.†

The only theoretical results not covered in this section refer to the Gilmore and Gomory[15] distance measure. These will be presented later with their algorithm.

Methods of Solution—General Comments and Classification

The methods for solving the traveling salesman problem usually can be divided into three basic parts: *a starting point, a solution generation scheme,* and *a termination rule.* When the termination rule is such that the iteration stops if and only if a tour is optimal, the method is *exact.* When the

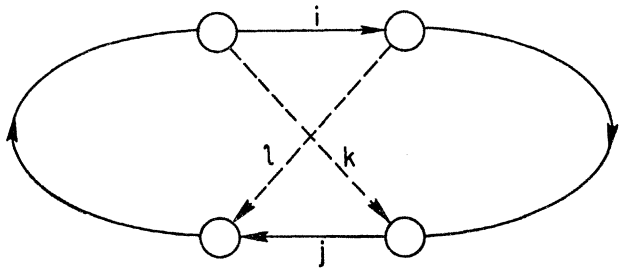

Fig. 2. Interchanging two arcs.

termination rule is such that the iteration stops if but not only if a tour is optimal, the method is *approximate.* In approximate methods the tour reached at termination generally depends on the starting point, so it is possible to produce many final tours by using different starting points. The best of these final tours is then selected.

Consider the following two termination rules:

(i) terminate if a tour t^0 has been found such that $z(t^0) = L$, where L is a lower bound on the length of all tours.

(ii) terminate if a tour $t^* = (i_1, i_2, \cdots, i_n, i_1)$ has been found such that $z(t^*) \leqq z(t)$ for all tours t that can be produced by interchanging two elements of t^*.

Clearly (i) is exact, that is t^0 is an optimal tour. However, t^* is best only in the local sense described by (ii). However, if t^* is optimal, then certainly we would terminate at t^* under rule (ii).

Since most starting points and termination rules depend, in part, on the

† A different longest path formulation is given by SALZ.[34, 35].

solution generation scheme, we will classify according to the solution generation method. We know of three fundamentally different ways of generating solutions.

A. Tour-to-Tour Improvement

The starting point is an arbitrary tour, say $t_0 = (1, 2, 3, \cdots, n, 1)$. The solution generation scheme is a rule for finding a better tour that is a 'neighbor' of the present tour. For example, t_1 is the best of the tours that can be generated by interchanging any element $i = 2, \cdots, n$ with 1. If $t_1 = (p, 2, 3, \cdots, n, p)$ then t_2 is the best tour that can be generated by interchanging $i = 1, \cdots, n$, $i \neq p$, with p. A termination rule could be to stop whenever no improvement can be made. We could then choose any other t_0 and repeat the iterative scheme or apply another, more sophisticated, scheme.

All procedures of this type known to us[3, 7, 25, 31, 32] are approximate. Each must be judged purely on its computational efficiency; i.e., quality of solution vs. time spent. Good results have been produced by iterative schemes developed by LIN,[25] and REITER AND SHERMAN.[31]

B. Tour Building

The starting point is an arbitrary node, say i_1. From i, we build a sequence $(i_1, i_2, \cdots, i_k)$ by successively including other nodes into the sequence. The procedure terminates when a tour is achieved. A very simple scheme of this type is the 'nearest neighbor' rule. From i_1 proceed to the nearest node i_2, from i_2 proceed to the nearest node not yet reached (not i_1 or i_2), $\cdots$, from i_n return to i_1.

Clearly, this method of tour building is approximate and there are many variations of the 'nearest neighbor' rule.[8, 21, 37] A method developed by KARG AND THOMPSON[21] has yielded good computational results. Exact tour building algorithms are dynamic programming[5, 17, 20] and the 'branch-and-bound'[24] algorithms of LITTLE ET AL.,[26] and HATFIELD AND PIERCE.[19]

C. Subtour Elimination

The starting point is an optimal solution to the assignment problem under the matrix C. If the solution to the assignment problem is a tour, it is optimal for the traveling salesman problem. If the optimal solution to the assignment problem is not a tour, an iterative scheme is used to eliminate subtours. Exact subtour elimination methods are integer linear programming,[6, 9, 10, 28-30] 'branch-and-bound' of EASTMAN[12] and SHAPIRO,[36] and the GILMORE-GOMORY method.[15]

DETAILED DESCRIPTIONS OF EFFECTIVE ALGORITHMS

IN THIS SECTION we present in some detail several algorithms that have produced impressive results. These include dynamic programming, integer programming, branch-and-bound, tour-to-tour approximations, and the Gilmore-Gomory method. In addition, some methods for reducing the size of problems will be discussed.

A. Dynamic Programming

Dynamic programming algorithms have been developed by BELLMAN,[5] GONZALES,[17] and HELD AND KARP.[20] The principle of optimality, as applied to the traveling salesman problem, yields Theorem 8.

Specifically, let $f_{k-1}(i_m|i_1, \cdots, i_{m-1}, i_{m+1}, \cdots, i_{k-1})$ be the length of a shortest path that starts at node 1, passes through $(i_1, \cdots, i_{m-1}, \cdots, i_{k-1})$ and terminates at node i_m.

From Theorem 8, it follows that a shortest partial tour from node 1 to node j that passes through $i_1, \cdots, i_{k-1}$ may be determined from

$$f_k(j|i_1, \cdots, i_{k-1}) = \min_{m=1, \ldots, k-1} [f_{k-1}(i_m|i_1, \cdots, i_{m-1}, i_{m+1}, \cdots, i_{k-1}) + c_{i_m j}].$$

Applying the above equation recursively we begin with

$$f_2(j|i_1) = c_{1i_1} + c_{i_1 j}, \text{ for all } i_1 \text{ and } j \neq 1, \text{ and } i_1 \neq j$$

and terminate with an optimal tour by solving

$$f_n(1|i_1, \cdots, i_{n-1}) = \min_{m=1, \ldots, n-1} [f_{n-1}(i_m|i_1, \cdots, i_{m-1}, i_{m+1}, \cdots, i_{n-1}) + c_{i_m 1}].$$

The difficulty in solving the recursive equations on a digital computer is the storage requirements. To compute f_{k+1}, we must have all values of f_k readily available in core storage. Once f_{k+1} has been calculated, f_k may be discarded. Furthermore, with an insignificant increase in computation time, the elements of f_{k+1} may be put in auxiliary storage as they are computed and then returned to core for the calculation of f_{k+2}. All other core storage requirements (program, data, etc.) are negligible compared to the core storage needed for f_k. The number of values of f_k is

$$g(n, k) = (n-1)!/(k-1)!(n-k-1)!.$$

The storage bottleneck occurs halfway through the computations. Since $g(15, 8) = 24{,}024$, the largest problem solvable on a machine with a $32k$ core (such as the IBM 7094) is $n = 15$.†

Let p be a bottleneck stage. By calculating the function f_p separately for suitable subsets of the combinations of argument values, storage demands can be reduced.[17] The procedure is to calculate only those values

† Held and Karp[20] report a code for the IBM 7090 for $n = 13$; however, auxiliary storage was not used for storing f_{k+1}.

of f_k, $k=1, \cdots, p-1$ needed to determine $f_p(j|i_2, \cdots, i_p)$ for fixed $i_2, \cdots, i_p$ and all $n-p$ values of j. Once the $n-p$ values of f_p are determined, we begin again with f_2 and calculate another $n-p$ values. When enough values of f_p are available $(n-p-1)$ values of f_{p+1} are calculated. We believe that an eighteen-city problem can be solved in this manner on a $32k$ core computer.

For a symmetric problem, with n even,† it is only necessary to compute $f_{n/2}$.[17] In particular

$$f_{n/2}(j|i_1, \cdots, i_{n/2-1})+f_{n/2}(j|i_{n/2+1}, \cdots, i_{n-1})$$

is the length of a tour that is minimum over all tours that proceed from node 1 through $(i_1, \cdots, i_{n/2-1})$ in some order, then to node j and back to node 1 through $(i_{n/2+1}, \cdots, i_{n-1})$ in some order. Consequently, by appropriately adding the $f_{n/2}$ values two at a time, an optimal tour can be established.

B. Integer Programming

The difficulties in finding an optimal tour in solving the integer program of Theorem 9 are the enormous number of loop constraints $(2^{n-1}-1)$ and the requirement that the $(n^2-n)/2$ variables x_{ij} equal 0 or 1 for symmetric distances. The solution of a linear program with the loop constraints and $0 \leqq x_{ij} \leqq 1$ generally will not satisfy $x_{ij}=0$ or 1.

However, in 1954 DANTZIG, FULKERSON, AND JOHNSON[9] found an optimal solution to a 42-city problem using this formulation.‡ They overcame the large number of loop constraints by beginning with only a few, and then adding new ones only as they were needed to block subtours. Combinatorial arguments were used to eliminate fractional solutions and to find an optimal tour. Finally, it was demonstrated that for the problem at hand, an ordinary linear program could be devised whose solution gave integer valued x_{ij}'s representing the optimal tour. The constraints that rule out some fractional solutions but no integer solutions were forerunners to GOMORY'S 'cutting plane' constraints for solving any integer linear program.[16]

After Gomory's method became available, MILLER, TUCKER, AND ZEMLIN[29] in 1960 experimented with solving traveling salesman problems using a 'cutting plane' algorithm and the formulation of Theorem 10. The results were rather disappointing.

The case was closed until 1966 when MARTIN[28] reported having solved Dantzig's 42-city problem by integer linear programming in less than five minutes on an IBM 7094. Unfortunately, this is the only large

† Minor modifications are required when n is odd; *see* reference 17.

‡ An elaboration of their original paper appears in reference 10.

problem for which Martin gives computational results. Thus, there is not enough evidence to draw any reasonable conclusions. Integer linear programming algorithms are notorious for converging rapidly on one problem and then performing miserably on the next.

Martin's impressive results, in contrast with Miller's can be attributed to any of a combination of several factors. Martin used the loop constraints of Theorem 9. The constraints of Theorem 9 appear to be better suited than the constraints of Theorem 10 for excluding fractional solutions. This can be seen by considering a simple example. Suppose we have a solution with the subtours (1, 2, 3, 1) and (4, 5, $\cdots$, n, 4). These subtours can be blocked by a Theorem 9 constraint with $S=\{1, 2, 3\}$, which is equivalent to

$$x_{12}+x_{23}+x_{31}\leqq 2.$$

Using the constraints of Theorem 10, three constraints are required

$$u_1-u_2+nx_{12}\leqq n-1,$$

$$u_2-u_3+nx_{23}\leqq n-1,$$

$$u_3-u_1+nx_{31}\leqq n-1.$$

When these three constraints are added we obtain

$$x_{12}+x_{23}+x_{31}\leqq 3-3/n.$$

Although this constraint is sufficient to block the subtour, it is weaker than

$$x_{12}+x_{23}+x_{31}\leqq 2,$$

and admits more feasible fractional solutions. Despite the fact that the formulation of Theorem 9 has many more constraints, generally very few of them will actually have to be used.

The tactics used to add the loop constraints can substantially effect the computation time. In solving the 42-city problem Martin began with 84 constraints, 42 of them coming from the assignment problem and 42 others being judiciously selected constraints in which S contained two nodes.

To eliminate fractional solutions Martin uses his "Accelerated Euclidean Algorithm:"[27] This algorithm employs Gomory's 'cutting planes,' but in a somewhat different manner than in Gomory's algorithm. It is possible that the Accelerated Euclidean Algorithm is very effective for traveling salesman problems. Finally, Martin's success is certainly, in part, attributable to the rapid developments in computer technology that have occurred in the past six years.

Other integer programming formulations have been obtained by Bock[6] and Mudrov.[30] But neither of these authors report computa-

tional results. FLEISHMANN[13] recently has applied BALAS'[2] 0-1 algorithm, but only to a seven-city problem. The results are discouraging, since the seven-city problem took more than five minutes on the IBM 7094.

C. Branch-and-Bound

Branch-and-bound algorithms have been developed by EASTMAN,[12] LITTLE, ET AL.,[26] and SHAPIRO.[36] Additionally, HATFIELD AND PIERCE[19] have used branch-and-bound algorithms to solve a job sequencing problem closely related to the traveling salesman problem, but further constrained because of job deadlines to be met. The work of Little, et al. is a tour-building algorithm, while the work of Eastman and Shapiro are examples of subtour elimination algorithms. The authors are not aware of a branch-and-bound algorithm based upon tour-to-tour improvement, although presumably one could be constructed. A rather complete survey of branch-and-bound methods has been given by LAWLER AND WOOD.[24]

The algorithm developed by Eastman and extended by Shapiro is a search technique in which one partitions the set of tours into subsets and calculates lower bounds on the cost of all tours in a subset. The initial bound is found by solving the associated assignment problem (Theorem 5). The bound is taken as the value of the solution to the assignment problem. If the solution to the assignment problem is not a feasible solution to the traveling salesman problem because of subtours, then one branches into k subproblems, where k is the number of arcs in one of the subtours. If the subtour is $(i_1, i_2, \cdots, i_k, i_1)$ then for subproblem 1 let $c_{i_1,i_2} = \infty$, for subproblem 2 let $c_{i_2,i_3} = \infty \cdots$ and for subproblem k let $c_{i_k,i_1} = \infty$. The subsets then are the set of all tours in which arc (i_1, i_2) is prohibited, etc. Shapiro chooses a subtour with smallest k for branching. This is intuitively appealing but is not necessarily the best choice. The k new assignment problems are solved and determine the lower bounds for all tours in their respective subsets. If any of these k solutions are tours and if the cost of one of these tours is less than or equal to the lower bounds on the other subsets, then that tour is optimal. If not, then one takes the subset with the lowest bound and branches again according to the subtours present in that solution. Eventually one is assured of finding an optimum tour.

Shapiro reports considerable difficulty with symmetric problems. Specifically, the number of subtours of length 2 is excessive. For this reason he adopts a different approach for symmetric problems. He takes the integer programming formulation of Theorem 9 (for the symmetric case) and initially uses n constraints, corresponding to the n ways in which set S may contain a single node. This guarantees that the solution will not have subtours of length 2. If the solution contains subtours he does

not add additional constraints in the spirit of Theorem 9, but instead partitions the problem into subproblems in the spirit of his work with asymmetric problems. The work appears unfinished at this time, but Shapiro provides some estimates of what can be expected when one deals with symmetric problems.

The algorithm developed by Little, et al. uses different tactics for branching and bounding. The calculation of bounds is based upon Theorems 6 and 7 and is referred to as matrix reduction. The reduced matrices are used for branching by partitioning the tours in a given subset into two subsets. This is done by committing an arc in one of the subsets and prohibiting that arc in the other subset. The computational experience of Shapiro appears to make using Little's algorithm less desirable.

D. Approximate Tour-to-Tour Improvement Algorithms

There are many approximate algorithms based upon tour-to-tour improvement. Most are minor variations of the two fundamental techniques discussed in this section. REITER AND SHERMAN[31] describe a series of 4 similar algorithms that culminate in their ALGO IV(r). For ALGO IV(1) one starts with a random tour, say $t=(i_1, i_2, \cdots, i_n)$ and finds the best tour that results from interchanging i_1 with i_2, then i_1 with i_3, $\cdots$, then i_1 with i_n [i.e., find the best position to insert i_1 in the sequence $(i_2, i_3, \cdots, i_n)$]. Denote this minimum tour as $(j_1, j_2, \cdots, j_n)$ and restart the procedure except using $(j_2, j_3, \cdots, j_n, j_1)$ (i.e., 'circulate' the minimum tour) as the initial tour. Each time a set of $(n-1)$ tours must be examined. The procedure is continued until n sets of $(n-1)$ tours are examined without finding a shorter tour.

ALGO IV(2) is an extension of ALGO IV(1). When ALGO IV(1) terminates, one applies ALGO IV(2) to the 'best' tour found by ALGO IV(1). ALGO IV(2) finds the best location for (i_1, i_2) in the sequence $(i_3, i_4, \cdots, i_n)$ and the best location for (i_2, i_1) in the sequence $(i_3, i_4, \cdots, i_n)$. Denote the best of these sequences as $(j_1, j_2, \cdots, j_n)$ and restart ALGO IV(2) except using $(j_2, j_3, \cdots, j_n, j_1)$ as the initial tour. ALGO IV(2) is reapplied until one obtains no improvement in n consecutive applications.

One now returns to ALGO IV(1). The entire process is repeated until ALGO IV(1) produces no improvement in n consecutive trials *and* ALGO IV(2) produces no improvement in n consecutive trials.

ALGO IV(3) is analogous except one finds the best location for (i_1, i_2, i_3) in the sequence $(i_4, i_5, \cdots, i_n)$ and the best location for (i_3, i_2, i_1) in the sequence $(i_4, i_5, \cdots, i_n)$. ALGO IV(r) finds the best location for $(i_1, i_2, \cdots, i_r)$ in the sequence $(i_{r+1}, i_{r+2}, \cdots, i_n)$ and the best location for $(i_r, i_{r-1}, \cdots, i_1)$ in the sequence $(i_{r+1}, i_{r+2}, \cdots, i_n)$.

A slightly different approach was suggested by LIN,[25] in which he finds approximate solutions which he calls λ optimal, for $\lambda = 2, 3, \cdots, n$. His λ is analogous to Reiter and Sherman's r, for $\lambda = 2$. One starts with a random tour say $t = (i_1, i_2, \cdots, i_n)$ and systematically tries to find a better tour by replacing 2 arcs of the tour by 2 other arcs.† Lin's algorithm is not constrained to changing the position of adjacent cities in the sequence. Figure 2 illustrates this procedure.

If the initial tour contained arcs i and j and if they are removed, then arcs k and l of Fig. 2 are uniquely determined. Note that one part of the original sequence will be visited in reverse order. A tour is '2-opt' when no improvement can be obtained by replacing any 2 arcs of the tour with 2 other arcs. Other 2-opt tours may be found by using different initial random tours.

Lin defined a tour as '3-opt' if one could not improve it by replacing any 3 arcs or any 2 arcs. Analogously, a tour is 'λ-opt' if it is '$(\lambda-1)$ opt' and if, in addition, one cannot improve the tour by replacing any λ arcs. He showed empirically that it is more efficient to find 3-opt tours than 2-opt tours. For a fixed amount of computer time, one would find fewer 3-opt tours than 2-opt tours, but the best 3-opt tour found would usually be at least as good and generally better than the best 2-opt tour found. He also found that it is not computationally efficient to find 4-opt tours.

Lin used a modified procedure in searching for 3-opt tours. After several 3-opt tours are found, any arc that appears in all 3-opt tours found is assumed to be in all other 3-opt tours and in this way a reduced problem is examined in all future computations.‡

E. Job Sequencing and the Gilmore-Gomory Algorithm[15]

An interesting application of the traveling salesman problem concerns the sequencing of n jobs on a machine. Assume there is a set-up cost of c_{ij} units if job j follows job i in the sequence and that the operation is cyclic—after the last job is done, the first is begun again. The objective is to minimize the sum of the set-up costs. This problem can be identified as a traveling salesman problem with node i corresponding to job i and the set-up cost c_{ij} corresponding to the distance between nodes i and j. Although, in general, the c_{ij} can be arbitrary real nonnegative numbers, Gilmore and Gomory have considered a particular distance measure that is quite meaningful for certain sequencing problems and have discovered a remarkably simple algorithm.

† Actually, CROES[7] first suggested this procedure and in addition reported a procedure for deriving exact solutions from these approximate solutions. His exact procedure is considered difficult to program for a computer.

‡ *See* Section F, "Partitioning and Decomposition."

Let

$$c_{ij}=\int_{B_i}^{A_j} f(x)\,dx, \qquad (A_j \geqq B_i)$$

$$c_{ij}=\int_{A_j}^{B_i} g(x)\,dx, \qquad (B_i \geqq A_j)$$

where (A_i, B_i) are arbitrary real numbers associated with job i, $i=1, \cdots, n$ and $f(x)$, $g(x)$ are any integrable functions satisfying

$$f(x)+g(x) \geqq 0.$$

The motivation for defining c_{ij} in this manner is that to start job i, the machine must be in state A_i and when job i is done the machine is in state B_i. Thus, if job j follows job i the state of the machine must be changed from B_i to A_j. It is assumed that the status of the machine can be described by one-state variable.

The first step in the algorithm is to solve the assignment problem. This is done very easily. In particular, assume the nodes are numbered so that $j>i$ implies $B_j \geqq B_i$. Now arrange the A_i so that $A_{i_1} \leqq A_{i_2} \leqq \cdots A_{i_n}$. The optimal solution to the assignment problem is $x_{ji_j}=1$, $j=1, \cdots, n$, $x_{ij}=0$ otherwise. This solution is not, in general, a tour.

By a series of interchanges the optimal assignment solution is converted into an optimal tour. An interchange α_{ps} applied to a solution $\{x_{ij}\}$ with $x_{pk}=x_{sm}=1$ yields the solution $\{x'_{ij}\}$, $x'_{pm}=x'_{sk}=1$, $x'_{pk}=x'_{sm}=0$, $x'_{ij}=x_{ij}$ otherwise. The interchanges are chosen specifically to remove subtours. Suppose s_1 and s_2 are subtours in a solution and α_{ij} is an interchange with $i \epsilon s_1$ and $j \epsilon s_2$, then the new solution contains all of the original subtours except that s_1 and s_2 have been replaced by a single subtour containing all their nodes. By systematically applying a sequence of these interchanges the assignment solution is transformed into a minimal tour.

F. Partitioning and Decomposition

The size of a traveling salesman problem can be reduced by imposing restrictions on the order in which the nodes can be traversed. Suppose the nodes $(i_1, \cdots, i_{n-k}) \epsilon S$ must be traversed consecutively in the given order. Then the original n node problem can be reduced to a $k+1$ node problem. Specifically the nodes are all of original nodes in $\bar{S}$ $[S \cup \bar{S}=\{1, 2, \cdots, n\}]$, plus one additional node θ. The distance matrix is C' with

$$c'_{ij}=c_{ij}, \qquad (i,j \epsilon \bar{S})$$

$$c'_{\theta j}=c_{i_{n-k}j}, \qquad (j \epsilon \bar{S})$$

$$c'_{i\theta}=c_{ii_1}. \qquad (i \epsilon \bar{S})$$

If an optimal tour under C' is $(i_\theta, i_{n-k+1}, \cdots, i_n)$, the original problem has an optimal tour $(i_1, \cdots, i_{n-k}, i_{n-k+1}, \cdots, i_n)$.

ROTHKOPF[33] shows another possible simplification. He is motivated by sequencing problems in which the cost of processing a job depends only on its class and the class of the previous job. He establishes that if for any class of nodes S

$$\begin{aligned} c_{ij} &= \alpha, && (i, j \epsilon S) \\ c_{ik} &= c_{jk}, && (i, j \epsilon S, k \epsilon \bar{S}) \\ c_{ki} &= c_{kj}, && (i, j \epsilon S, k \epsilon \bar{S}) \end{aligned}$$

$$\alpha + c_{pq} < c_{pj} + c_{jq} \quad \text{for all} \quad j \epsilon S, p, q \epsilon \bar{S},$$

then all of the nodes in S are traversed consecutively (in any order) in an optimal tour.

HELD AND KARP[20] use partitioning to obtain approximate solutions. Following reference 20 suppose we have some tour $t=(i_1, i_2, \cdots, i_n, i_1)$. Assume that n is too large to solve the problem exactly, but that $k+1<n$ is small enough to apply an exact method. Define a $k+1$ city problem by partitioning the nodes. For example, we might assume that $i_1, i_2, \cdots, i_{n-k}$ is a segment of an optimal tour and consider a $k+1$ node problem with distance matrix C' as described above. Or we could consider the $k+1$ city problem derived by assuming the arcs (i_1, i_2), $(i_3, i_4), \cdots, (i_{2(n-k)-3}, i_{2(n-k)-2})$ are segments of an optimal tour. In any case, whatever partition is chosen, the tour t' obtained from the optimal tour for the $k+1$ city problem is such that

$$z(t') \leqq z(t).$$

We can then partition t' and continue.

Held and Karp give some rules for selecting 'good' partitions. The rules attempt to identify a partition such that an optimal tour on the partitioned problem is as short as possible. The partitioned problems ($n \leqq 13$) are solved by dynamic programming, although other exact methods could be used.

KARG AND THOMPSON[21] use partitions in a slightly different way. Assume that $i_1, \cdots, i_k$ appears promising as an optimal tour segment. Consider two traveling salesman problems P_1 and P_2. P_1 contains the k nodes $(i_1, \cdots, i_k)$ and the restriction that the arc (i_k, i_1) must be traversed and P_2 contains the $n-k+2$ nodes $(i_1, i_k, i_{k+1}, \cdots, i_n)$ with the restriction that (i_1, i_k) must be traversed. Suppose the solutions to P_1 and P_2 are respectively $(i_1, \cdots, i_k, i_1)$ and $(i_k, i_{k+1}, \cdots, i_n, i_1, i_k)$. The optimal tours are combined as $(i_1, i_2, \cdots, i_k, i_{k+1}, \cdots, i_n, i_1)$. Karg and Thompson select their tour segment $(i_1, \cdots, i_k)$ using the criterion that the closed

circuit $(i_1, i_2, \cdots, i_k, i_1)$ approximates a convex set in two-dimensional Euclidean space. The test for convexity (not really convexity in the strict mathematical sense) is algebraic and they claim it has been applied to non-Euclidean problems with some success as well. Subproblems such as P_2 once solved can be partitioned further, based upon the notion of convexity. Karg and Thompson use their tour building heuristic to solve the partitioned problems, although it would seem more desirable to apply an exact procedure when the subproblems are small enough.

LIN[25] and ROBERTS AND FLORES[32] reduce problem size by assuming that once an arc appears in enough solutions obtained by approximate algorithms it will be in an optimal tour.

G. Other Approaches

A review of the traveling salesman problem through approximately 1960 appeared in reference 1. BEARDWOOD, HALTON, AND HAMMERSLY[4] derived asymptotic bounds on the length of an optimal tour containing a large number of cities contained in a region of specified dimensions. KRUSKAL[22] pointed out a possible relation between the traveling salesman and shortest spanning tree problems. DERMAN AND KLEIN[11] consider an inspection and maintenance problem for which the model is a traveling salesman problem without the 0-1 restrictions on the variables. LAWLER[23] establishes that the traveling salesman problem is a special case of the quadratic assignment problem, which is a combinatorial problem even more difficult than the traveling salesman problem itself.

COMPUTATIONAL EXPERIENCE

COMPARISON OF published computational experience is always difficult because of the different machines used and relative efficiency of different programming languages. The majority of reported times are for the IBM 1620 class and the IBM 7090 class of computing machines. The authors' personal experience on these two classes of machines indicate that as a first order approximation the 7090 is from 50 to 300 times faster than the 1620. Obviously, the exact ratio depends upon the particular application.

Table I summarizes the reported computational time for exact algorithms. In examining this table, one should be aware that the times reported for integer programming and branch-and-bound are expected times, and particular problems may take considerably longer.

It is difficult to compare approximate algorithms since one is interested in the probability of finding optimal tours and the amount of computer time required. Ideally one would compare algorithms in terms of the best

solution found for a particular problem as a function of the total computer time required. The variance of reporting methods used in the literature makes such comparisons impossible.

Two well-known problems that have been treated by most approximate algorithms are the 48-city problem proposed by HELD AND KARP[20] and the 57-city problem proposed by KARG AND THOMPSON.[21] Optimum

TABLE I

COMPUTATIONAL EXPERIENCE (EXACT ALGORITHMS)

	Largest problem solved	Computer	Time (min)	Remarks
Dynamic programming (Held and Karp)[20]	13	7090	0.28	(1) Storage limitations prevented the solution of larger problems. (2) Required computer time is deterministic.
Branch-and-Bound (Little, et al)[26]	40	7090	8.37	(1) Time reported is the expected time and one might experience large deviations for a particular problem. (2) Time reported refers to randomly selected, asymmetric c_{ij}. Considerable difficulty reported for Euclidean problems.
Branch-and-bound (Shapiro)[30]	70 40	1620 1620	103.5 8.16	(1) Only one 70-city problem was solved. The actual computer time for a particular problem might be significantly different; however, this is the largest problem reported solved by any exact algorithm. (2) Time reported refers to randomly selected asymmetric c_{ij}. Considerable difficulty reported for randomly selected symmetric c_{ij}. (3) 40-city result is given to aid direct comparison with Little's experience (note the difference in machine used).
Integer programming (Martin)[28]	42	7094	Approx. 5 min.	(1) The time was reported for only one problem. (2) Considerable variance may be experienced with particular problems. (3) Other authors have reported much less favorable results from integer programming algorithms.

solutions are not known with certainty for either problem. Table II compares five approximate algorithms with respect to these problems. The algorithms of LIN[25] and REITER and SHERMAN[31] were both able to find the best known solutions to both problems. We feel that the computational efficiency of these two algorithms is similar.

Lin reports that the average IBM 7094 Model II computer time to obtain a 3-opt tour is under $30n^3$ microseconds and the probability that a '3-opt' tour is actually optimum is approximately $p(n) = 2^{-n/10}$. He suggests that a pessimistic estimate of this probability is $\frac{1}{4}\,p(n)$. Based

upon empirical results, the computer time required to achieve 99 per cent probability of finding the optimal solution for a 40-city problem is 2.5 min and for a 60-city problem, 8 min. Lin estimates that in 100 min, there is a 54 per cent probability of solving a 100-city problem. He has used this method to treat a 105-city problem.

CONCLUSION

If the authors were faced with the problem of finding a solution to a particular traveling salesman problem we would use dynamic programming for problems with 13 cities† or less, Shapiro's branch-and-bound algorithm for larger problems (up to about 70–100 cities for asymmetric

TABLE II

Computational Experience (Approximate Algorithms)

Algorithm	Length of Best Tour Found	
	48 City (Held and Karp[20])	57 City (Karg and Thompson[21])
Held and Karp[20] (Dynamic programming with partitioning)	11470	—
Karg and Thompson[21]	—	12985
Reiter and Sherman[31]	11461	12955
Shen Lin[25]	11461	12955
Roberts and Flores[32]	11461	12985
Best known solution	11461	12955

problems and up to about 40 cities for symmetric problems) and Shen Lin's '3-opt' algorithm for problems that cannot be handled by Shapiro's algorithm. We recommend dynamic programming over branch-and-bound for smaller problems because, although the expected computer time might be greater, we are assured that the maximum time is very small.

REFERENCES

1. Arnoff, E. L. and S. S. Sengupta, "The Traveling Salesman Problem," *Progress In Operations Research*, Vol. I (R. L. Ackoff, ed.) 150–157, Wiley, New York, 1961.
2. Balas, E., "An Additive Algorithm for Solving Linear Programs with Zero-One Variables," *Opns. Res.* **13**, 517–546 (1965).

† With the judicious use of auxiliary storage, one should be able to solve slightly larger problems by dynamic programming.

3. BARACHET, L. L., "Graphic Solution of the Traveling Salesman Problem," *Opns. Res.* **5**, 841–845 (1957).
4. BEARDWOOD, J., J. H. HALTON, AND J. M. HAMMERSLEY, "The Shortest Path Through Many Points," *Proc. Cambridge Phil. Soc.*, (Math. and Phys. Sci.) **55**, 299–327 (1959).
5. BELLMAN, R., "Dynamic Programming Treatment of the Traveling Salesman Problem," *J. ACM* **9**, 61–63 (1962).
6. BOCK, F., "Mathematical Programming Solution of Traveling Salesman Examples," *Recent Advances in Mathematical Programming*, (R. L. GRAVES AND P. WOLFE, eds.) 339–341 McGraw-Hill, New York, 1963.
7. CROES, G. A., "A Method for Solving Traveling Salesman Problems," *Opns. Res.* **6**, 791–812 (1958).
8. DACEY, M. F., "Selection of An Initial Solution for the Traveling Salesman Problem," *Opns. Res.* **8**, 133–134 (1960).
9. DANTZIG, G. B., D. R. FULKERSON, AND S. M. JOHNSON, "Solution of a Large Scale Traveling Salesman Problem," *Opns. Res.* **2**, 393–410 (1954).
10. ———, "On a Linear Programming, Combinatorial Approach to the Traveling Salesman Problem," *Opns. Res.* **7**, 58–66 (1959).
11. DERMAN, C. AND M. KLEIN, "Surveillance of Multicomponent Systems: A Stochastic Traveling Salesman's Problem," *Nav. Res. Log. Quart.* **13**, 103–112 (1966).
12. EASTMAN, W. L., "Linear Programming with Pattern Constraints," Ph.D. Dissertation, Harvard, 1958.
13. FLEISHMANN, B., "Computational Experience with an Algorithm of Balas," *Opns. Res* **15**, 153-155 (1967.)
14. FLOOD, M. M., "The Traveling Salesman Problem," *Opns. Res.* **4**, 61–75 (1956).
15. GILMORE, P. C., AND R. E. GOMORY, "Sequencing a One-State Variable Machine: A Solvable Case of the Traveling Salesman Problem," *Opns. Res.* **12**, 655–679 (1964).
16. GOMORY, R. E., "An Algorithm for Integer Solutions to Linear Programs," *Recent Advances in Mathematical Programming* (R. L. GRAVES AND P. WOLFE, eds.) 269–302, McGraw-Hill, New York, 1963.
17. GONZALES, R. H., "Solution to the Traveling Salesman Problem by Dynamic Programming on the Hypercube," Tech. Rep. No. 18, O.R. Center, M.I.T., 1962.
18. HARDGRAVE, W. W. AND G. L. NEMHAUSER, "On the Relation Between the Traveling Salesman and the Longest Path Problem," *Opns. Res.* **10**, 647–657 (1962).
19. HATFIELD, D. J. AND J. F. PIERCE, "Production Sequencing by Combinatorial Programming," IBM Cambridge Scientific Center, Cambridge, Mass., 1966.
20. HELD, M. AND R. M. KARP, "A Dynamic Programming Approach to Sequencing Problems," *SIAM* **10**, 196–210 (1962).
21. KARG, L. L. AND G. L. THOMPSON, "A Heuristic Approach to Solving Traveling Salesman Problems," *Management Sci.* **10**, 225–248 (1964).

22. KRUSKAL, J. B., "On the Shortest Spanning Subtree of a Graph and the Traveling Salesman Problem," *Proc. Amer. Math. Soc.* **2**, 48–50 (1956).
23. LAWLER, E. L., "The Quadratic Assignment Problem," *Management Sci.* **7**, 586–599 (1963).
24. ———, AND D. E. WOOD, "Branch-and-Bound Methods: A Survey," *Opns. Res.* **14**, 699–719 (1966).
25. LIN, S., "Computer Solution of the Traveling Salesman Problem," *Bell System Tech. J.* **44**, 2245–2269 (1965).
26. LITTLE, J. D. C., K. G. MURTY, D. W. SWEENEY, AND C. KAREL, "An Algorithm for the Traveling Salesman Problem," *Opns. Res.* **11**, 979–989 (1963).
27. MARTIN, G. T., "An Accelerated Euclidean Algorithm for Integer Linear Programming," *Recent Advances in Mathematical Programming* (R. L. GRAVES AND P. WOLFE, eds.), 311–318, 1963.
28. ———, "Solving the Traveling Salesman Problem by Integer Linear Programming," *CEIR*, New York, 1966.
29. MILLER, C. E., A. W. TUCKER, AND R. A. ZEMLIN, "Integer Programming Formulations and Traveling Salesman Problems," *J. ACM* **7**, 326–329 (1960).
30. MUDROV, V. I., "A Method of Solution of the Traveling Salesman Problem by Means of Integer Linear Programming (The Problem of Finding the Hamiltonian Paths of Shortest Length in a Complete Graph)" *IAOR* **5**, Abstract 3330, 1965.
31. REITER, S. AND G. SHERMAN, "Discrete Optimizing," *SIAM* **13**, 864–889 (1965).
32. ROBERTS, S. M. AND B. FLORES, "An Engineering Approach to the Traveling Salesman Problem," *Management Sci.* **13**, 269–288 (1966).
33. ROTHKOPF, M., "The Traveling Salesman Problem: On the Reduction of Certain Large Problems to Smaller Ones," *Opns. Res.* **14**, 532–533 (1966).
34. SALZ, N. P., "The NORMA: A Possible Basis for a Theory for the Traveling Salesman Problem," Cornell Aero. Lab., Buffalo, N.Y., 1964.
35. ———, "The NORMA: A Theory for the Traveling Salesman Problem," Cornell Aero. Lab., Buffalo, N.Y., 1966.
36. SHAPIRO, D., "Algorithms for the Solution of the Optimal Cost Traveling Salesman Problem," Sc.D. Thesis, Washington University, St. Louis, 1966.
37. STOUDHAMMER, J. AND M. ASH, "A Sufficiency Solution of the Traveling Salesman Problem," Sys. Aer. Corp., Santa Monica, California, 1966.