Génie Industriel

Rapport Technique no. EP78-R-28

THE TRAVELING SALESMAN PROBLEM AS A CONSTRAINED SHORTEST PATH PROBLEM: THEORY AND COMPUTATIONAL EXPERIENCE

by

David J. HOUCK jr. University of Maryland Baltimore County, Baltimore, Md, USA
Jean-Claude PICARD Ecole Polytechnique de Montréal
Maurice QUEYRANNE Ecole Polytechnique de Montréal
and
R.R. VEMUGANTI University of Baltimore Baltimore, Md, USA

SEPTEMBER 1978

Ecole Polytechnique de Montréal

Campus de l'Université de Montréal
Case postale 6079
Succursale 'A'
Montréal, Québec
H3C 3A7

THE TRAVELING SALESMAN PROBLEM AS A CONSTRAINED SHORTEST PATH PROBLEM: THEORY AND COMPUTATIONAL EXPERIENCE

by

David J. HOUCK jr.	University of Maryland Baltimore County, Baltimore, Md, USA
Jean-Claude PICARD	Ecole Polytechnique de Montréal
Maurice QUEYRANNE and R.R. VEMUGANTI	Ecole Polytechnique de Montréal University of Baltimore, Baltimore, Md, USA.

SEPTEMBER 1978

ABSTRACT

There are many relaxations of the traveling salesman problem, most notably the assignment problem and the 1-tree and 1-arborescence relaxations of Held and Karp. In this paper we present a new relaxation, called an n-path which can be incorporated in the general framework of Held and Karp's algorithm. An n-path is a path from a given vertex to itself which contains exactly n arcs. A set of vertex penalties which sum to zero may change the length of n-paths which are not tours but does not change the length of tours. It is shown that the problem of finding optimal vertex penalties is related to the linear programming relaxations of some linear integer programming formulations of the traveling salesman problem. Further it is shown that the n-path relaxation using the optimal vertex penalties uniformly provides a bound better than the assignment relaxation. Computational experience based on two branch and bound algorithms, incorporating the n-path relaxation is discussed.

INTRODUCTION

The literature on the traveling salesman problem (TSP) is rather extensive [2,3,4,7,14,16,17,21,25,35,36]. But a good algorithm as defined by Edmonds [8] has not yet been found. The results of reference 22 strongly indicate that such an algorithm may not exist. Most of the exact algorithms [3,4,16,17,25,35,36] to date are of the branch and bound variety. These algorithms are based on the relaxations of TSP for which good algorithms have been found. The relaxations of TSP so far addressed in the literature are assignment [4,35,36], 2-matching [4], 1-tree [16,17] and 1-arborescence [16,17]. Held and Karp [16,17] have presented impressive computational experience using the 1-tree relaxation for the symmetric TSP. Smith, Srinivasan and Thompson [36] have presented superior computational experience using the assignment relaxation for the asymmetric TSP.

The efficiency of any branch and bound algorithm for the TSP depends upon the bounds generated by the relaxations and the time required to calculate those bounds. In section 2 of this paper we present a new relaxation of the TSP called an n-path. Vertex penalties which sum to zero may change the length of n-paths which are not tours but the length of tours will not be changed. Determining the optimal vertex penalties which will minimize the difference between the length of an optimal tour and the optimal n-path is similar to determining the optimal vertex penalties using the 1-tree relaxation. Therefore our work is closely related to that of Held and Karp. In section 3 it is shown that the bounds generated by the n-path relaxation using the optimal vertex penalties are uniformly better than the bounds generated by the assignment relaxation.

Section 4 discusses some properties of optimal penalties while Section 5 shows the relation between n-paths and an integer programming formulation of the TSP given by Hadley [15]. Section 6 illustrates how subgradient

optimization can be used to find good vertex penalties and in section 7 this technique is incorporated into two different branch and bound algorithms. Section 8 develops a refinement to the n-path problem improving the bounds generated and section 9 presents computational experience with the two algorithms.

2. N-PATHS AND RELATED LINEAR INTEGER PROGRAMMING FORMULATIONS OF TSP

Consider a directed complete graph G with n vertices and distances d_{ij} from vertex i to j, ($d_{ii} = \infty$ for $i = 1,2,\ldots,n$ and any non-existant arcs can be given infinite length.) A k-path between vertices i and j (not necessarily distinct) is a sequence of k arcs $c = \{(i,i_1), (i_1,i_2),\ldots, (i_{k-1},j)\}$. A k-path c is elementary if the vertices $i, i_1,\ldots, i_{k-1}, j$ are all distinct except possibly i and j. The length d_c of a k-path c is given by the sum of the lengths of the arcs in c. It is obvious that an n-path from vertex 1 to vertex 1 is a tour if and only if it is elementary. An optimal solution to TSP can be can be obtained by finding an elementary n-path of minimum length from vertex 1 to itself. This is a difficult problem. However, dynamic programming [29,33,34] can be used to find an n-path (not necessarily elementary) from vertex 1 to itself of minimum length. Since we are interested in tours, it is possible to consider only n-paths which do not contain vertex 1 as an intermediate vertex. If we let $f(i,k)$ denote the minimum length of a k-path from vertex 1 to vertex i then $f(1,n)$ is the length of interest. The following recursive relations define $f(i,k)$.

$$f(i,1) = d_{1i} \quad \text{for} \quad i = 2,3,\ldots,n$$

$$f(i,k) = \min_{j\in\{2,\ldots,n\}} [f(j,k-1) + d_{ji}] \qquad \begin{matrix} i = 2,3,\ldots,n \\ k = 2,3,\ldots,n-1 \end{matrix}$$

and

$$f(1,n) = \min_{j\in\{2,\ldots,n\}} [f(j,n-1) + d_{j1}]$$

The n-path corresponding to $f(1,n)$ can be found by recording, for each $f(i,k)$,

the index j which was the minimum. It can be seen that $f(1,n)$ and the corresponding n-path can be found in $O(n^3)$ additions and comparisons. Obviously if the n-path is a tour it solves TSP.

A Linear Integer Programming Formulation of TSP.

Let C be the set of all n-paths from vertex 1 to itself which do not contain the vertex 1 as an intermediate vertex. For each $c \in C$, let d_c be the length of the n-path and let a_{ci} be the number of times the n-path c passes through vertex i. It is easy to verify that

$$\sum_{i=2}^{n} a_{ci} = n-1 \quad \text{for all} \quad c \in C . \tag{2.1}$$

The following is a linear integer programming formulation of the TSP.

(IP1) Minimize $\sum_{c \in C} d_c x_c$

s.t. $\sum_{c \in C} a_{ci} x_c = 1 , \quad i = 2,3,\ldots,n$

and $x_c \geq 0$ and integer.

Since all a_{ci} are nonnegative integers satisfying (2.1), an optimal solution to IP1 must satisfy $x_{c^*}=1$ for some $c^* \in C$ and $x_c = 0$ for all $c \in (C-c^*)$. It follows that the n-path c^* is an optimal tour. Let LP1 be the linear programming relaxation of IP1 and let P be the polyhedron defined by its constraints. Then we can prove the following interesting result which is not included in the results of Padberg and Rao [30].

Theorem 1. The diameter of P is two.

Proof: Since IP1 has a feasible solution, P is non-empty. Let B be a feasible basis of LP1. If B contains the column vector a_{c^*} associated with c^* then the basis B to LP1 yields the optimal solution

to IP1. Suppose the basis B does not contain the vector a_{c*}. The values of the basic variables associated with the basis B are given by

$$x_B = B^{-1}\,\underline{1}\ .$$

Noting that all elements of a_{c*} are unity we obtain

$$y_{c*} = B^{-1}a_{c*} = B^{-1}\underline{1} = x_B\ .$$

Since at least one element of x_B is positive, it follows that

$\text{Min}\left\{\frac{x_{Bi}}{y_{c*i}} \mid x_{Bi} > 0\right\} = 1$. Therefore, we can enter the column a_{c*} and remove any column r with $x_{Br} > 0$ from B, using the usual rules of pivoting. The new basis after this single pivot is the optimal solution to IP1 and hence adjacent to all extreme points of P. Thus the diameter of P is two. ∎

It is possible to develop a column generation scheme (see reference 23) to solve LP1. Let B be a basis to LP1 (the initial basis may consist of artificial variables) and let $\pi = (\pi_2, \pi_3, \ldots, \pi_n)$ be the corresponding dual variables. The shadow price of the variable x_c is

$$\hat{d}_c = d_c - \pi a_c$$

where a_c is the column vector associated with x_c It can be seen that $\hat{d}_c$ is the length of n-path c with penalties π_i associated with vertex i. Such a penalty may be enforced by modifying the distance matrix by $d_{ij} \leftarrow d_{ij} - \lambda\pi_i - (1-\lambda)\pi_j$ for each i,j; where λ is a fixed parameter satisfying $0 \le \lambda \le 1$. Thus we can find $\hat{d}_r = \underset{c \in C}{\text{Min}}\ \hat{d}_c$ by finding the shortest n-path using the modified distances. If $\hat{d}_r \ge 0$ then the current solution is optimal. Otherwise we select column a_r to enter the basis.

Lemma 1. If the column a_r to enter the basis corresponds to a tour then it is an optimal tour.

Proof: For any n-path c and its associated column a_c we have

$$\hat{d}_r = d_r - \pi a_r \leq d_c - \pi a_c .$$

If c is also a tour, then $\pi a_r = \pi a_c$ and hence $d_r \leq d_c$. ∎

The optimal solution to LP1 provides a lower bound on the length of an optimal tour. This information can be used in developing a branch and bound algorithm to solve IP1. However the column generation scheme is in general very slow in converging to the optimal solution and therefore only small size problems can be solved using LP1.

3. RELATION BETWEEN LP1 AND THE ASSIGNMENT RELAXATION

The assignment relaxation of TSP is:

(AR) Minimize $\sum_{i=1}^{n} \sum_{j=1}^{n} d_{ij} x_{ij}$

$$\text{s.t.} \quad \sum_{j=1}^{n} x_{ij} = 1 \qquad i = 1,2,\ldots,n$$

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

$$x_{ij} \geq 0 \qquad \text{for all } (i,j).$$

With the help of the following results it will be shown that LP1 always provides better bounds than AR. Consider any solution to LP1. Let L be the set of all n-paths in that solution, i.e., $L = \{c \in C | x_c > 0\}$. Let $p(i,j)$, the weights assigned to the arc (i,j), be given by

$$p(i,j) = \sum_{c \in L} x_c n_c(i,j) \tag{3.1}$$

where $n_c(i,j)$ is the number of times the n-path c passes through the arc (i,j). For any n-path c we have for each vertex i

$$\sum_{j=1}^{n} n_c(i,j) = \sum_{j=1}^{n} n_c(j,i) = a_{ci}$$

and

$$d_c = \sum_{i=1}^{n} \sum_{j=1}^{n} n_c(i,j) d_{ij} \tag{3.2}$$

Now

$$\sum_{j=1}^{n} p(i,j) = \sum_{c \in L} x_c \sum_{j=1}^{n} n_c(i,j)$$

$$= \sum_{c \in L} x_c a_{ci} = 1 \quad \text{for } i = 1,2,\ldots,n \tag{3.3}$$

Similarly it is easy to show that

$$\sum_{i=1}^{n} p(i,j) = 1 \quad \text{for } j = 1,2,\ldots,n \tag{3.4}$$

Theorem 2. Let z_{LP1} and z_{AR} be the optimal values of LP1 and AR respectively. Then for any graph $z_{LP1} \geq z_{AR}$.

Proof: Given any solution to LP1, from (3.3) and (3.4) it follows that p(i,j), as defined in (3.1), is feasible to AR. One value of the objective function associated with p(i,j) is

$$\sum_{i=1}^{n}\sum_{j=1}^{n} p(i,j)d_{ij} = \sum_{c\in L} x_c \sum_{i=1}^{n}\sum_{j=1}^{n} n_c(i,j)d_{ij}$$

$$= \sum_{c\in L} x_c d_c \quad \text{from (3.2).}$$

Thus corresponding to each solution to LP1, there exists a solution to AR with equal objective functions. Therefore $z_{LP1} \geq z_{AR}$. ∎

The following theorem specifies under which conditions the bound z_{LP1} is not strictly better than z_{AR}.

Theorem 3. If $z_{AR} = z_{LP1}$ then at least one of the following assertions hold.

(i) The optimal assignment is not unique.

(ii) The shortest tour is optimal to both LP1 and AR.

Proof: Suppose $z_{AR}=z_{LP1}$ and the optimal assignment is unique. Let p(i,j) be the weight assigned to the arc (i,j) using the optimal solution to LP1. Consider the graph $G_1 = (X,E)$ where $X=\{1,2,\ldots,n\}$ and $E = \{(i,j) \mid p(i,j) > 0\}$. From LP1 we know that in G_1 there exist paths from vertex 1 to every other vertex and vice versa. Therefore G_1 is strongly connected.

Since $z_{AR} = z_{LP1}$ we know that p(i,j) is optimal to AR . Since it is a unique optimal and strongly connected solution it must be a tour. ∎

The following section describes another linear program which is closely related to LP1.

4. OPTIMAL PENALTIES

Suppose we assign penalties $\pi_2, \pi_3, \ldots, \pi_n$ to the vertices of the graph G_1.

If we replace the distances d_{ij} by $d_{ij} - \lambda\pi_i - (1-\lambda)\pi_j$, as stated above, then the length of any n-path c is $d_c - \pi a_c$. If the penalties satisfy

$$\sum_{i=2}^{n} \pi_i = 0 \qquad (4.1)$$

then for any tour

$$d_c - \pi a_c = d_c .$$

However this is not necessarily true for an n-path which is not a tour. Let d_{c^*} be length of an optimal tour. We would like to choose π so that

$$d_{c^*} - \min_{c \in C} (d_c - \pi a_c)$$

is a minimum. This is equivalent to the following linear program.

LP2: Maximize ω

$$\text{s.t.} \quad \omega + \pi a_c \leq d_c \qquad \text{for all } c \in C$$

$$\text{and} \quad \sum_{i=2}^{n} \pi_i = 0$$

$$\omega, \pi_i \text{ are unrestricted .}$$

Lemma 2: The dual of LP2 is equivalent to LP1.

Proof: Let x_c be the dual variable associated with each constraint $c \in C$ and let y be the dual variable associated with the last constraint. Then the dual of LP2 is

$$\text{Minimize} \quad \sum_{c \in C} d_c x_c$$

$$\text{s.t.} \quad \sum_{c \in C} x_c = 1$$

$$\sum_{c \in C} a_{ci} x_c + y = 0 \qquad i = 2,3,\ldots,n \ .$$

$$x_c \geq 0 \text{ and } y \text{ unrestricted.}$$

Adding the second set of constraints we get

$$\sum_{c \in C} x_c \sum_{i=2}^{n} a_{ci} + (n-1)y = 0$$

Noting that $\sum_{i=2}^{n} a_{ci} = n - 1$ for all $c \in C$ and that $\sum_{c \in C} x_c = 1$,

we get $y = -1$. Therefore the above problem is equivalent to

$$\text{Minimize} \quad \sum_{c \in C} d_c x_c$$

$$\text{s.t.} \quad \sum_{c \in C} a_{ci} x_c = 1 \qquad i = 2,3,\ldots,n.$$

$$\text{and} \quad x_c \geq 0$$

This is just LP1. ∎

Theorem 4: The following two properties are equivalent

(i) There exist penalities π^* such that the optimal tour c^* is an n-path of minimum length.

(ii) There exists an integer optimal solution to LP1.

Proof: (ii) $\Rightarrow$ (i). If the optimal solution to LP1 is integer then $z_{LP1} = d_{c^*}$. Let π^* be the row vector of corresponding dual variables. Then we have $\pi^* \underline{1} = z_{LP1} = d_{c^*}$. By optimality of LP1 we have $\hat{d}_c = d_c - \pi^* a_c \geq 0$ for any n-path c. However $\hat{d}_{c^*} = d_{c^*} - \pi^* \underline{1} = 0$. Thus c^* is the shortest n-path with the weights π^*.

(i) $\Rightarrow$ (ii) . If the optimal tour c^* is not optimal to LP1 then we must have $z_{LP1} < d_{c^*}$.

For all penalties π, satisfying (4.1), the duality theorem implies

$$\omega(\pi) = \underset{c \in C}{\text{Min}} \ (d_c - \pi a_c) \leq z_{LP1} < d_{c^*} . \qquad (4.2)$$

Let π^* be the penalties which yield the tour c^* and let $p = \pi^* \underline{1}$ and $\pi' = \pi^* - (p/(n-1))\underline{1}$. It is easy to verify that $\omega(\pi') = \omega(\pi^*) + p = d_{c^*}$. This contradicts (4.2). Therefore the optimal tour c^* is an optimal solution to LP1. ∎

Observe that LP2 has a large number of constraints. An optimal or near optimal solution to this problem can be obtained using a sub-gradient algorithm [14,17,19]. This is discussed in a later section. There is yet another linear integer programming formulation of TSP which is related to LP2 and is addressed in the following section.

5. ANOTHER LINEAR INTEGER PROGRAMMING FORMULATION OF TSP

The following integer programming formulation of TSP is given in Hadley [15]. Let

$$x_{ijk} = \begin{cases} 1 & \text{if the kth arc of the tour goes from vertex i to vertex j} \\ 0 & \text{otherwise .} \end{cases}$$

(IP2) Minimize $\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n} d_{ij}\, x_{ijk}$

s.t.
$$\sum_{j=2}^{n} x_{1j1} = 1 \tag{5.1}$$

$$x_{1j1} = \sum_{m=2}^{n} x_{jm2} \qquad j=2,\ldots,n \tag{5.2}$$

$$\sum_{i=2}^{n} x_{ijk} = \sum_{m=2}^{n} x_{jmk+1} \qquad j=2,\ldots,n \text{ and } k=2,\ldots,n-2 \tag{5.3}$$

$$\sum_{i=2}^{n} x_{ijn-1} = x_{j1n} \qquad j=2,\ldots,n \tag{5.4}$$

$$x_{1j1} + \sum_{i=2}^{n}\sum_{k=2}^{n-1} x_{ijk} = 1 \qquad j=2,\ldots,n \tag{5.5}$$

$$\sum_{i=2}^{n} x_{i1n} = 1 \tag{5.6}$$

$$\sum_{i=2}^{n}\sum_{j=2}^{n} x_{ijk} = 1 \qquad k=2,\ldots,n-1 \tag{5.7}$$

and x_{ijk} is 0 or 1 for all i,j,k.

Constraints (5.1) and (5.6) insure that vertex 1 is the first and last in the tour. Constraints (5.2)-(5.4) insure that if we go to vertex j on the kth arc, we must leave vertex j on the (k+1)st arc. Constraints (5.5) and (5.7) insure that we visit each vertex and that we go to some vertex on each leg of the tour.

Lemma 3: Constraints (5.6) and (5.7) are redundant.

Proof: Summing the constraints in (5.2), (5.3), and (5.4) over j we obtain

$$\sum_{j=2}^{n} x_{1j1} = \sum_{j=2}^{n} \sum_{m=2}^{n} x_{jm2},$$

$$\sum_{j=2}^{n} \sum_{m=2}^{n} x_{jmk} = \sum_{j=2}^{n} \sum_{m=2}^{n} x_{jmk+1}, \qquad k=2,\ldots,n-2$$

and $$\sum_{j=2}^{n} \sum_{m=2}^{n} x_{jmn-1} = \sum_{j=2}^{n} x_{j1n} \quad .$$

From this and (5.1) we obtain

$$\sum_{j=2}^{n} \sum_{m=2}^{n} x_{jmk} = 1, \qquad k=2,\ldots,n-1$$

and

$$\sum_{j=2}^{n} x_{j1n} = 1$$

and therefore the constraints (5.6) and (5.7) are redundant. ∎

Let LP3 be the linear programming relaxation of IP2 (obtained by replacing the 0-1 restrictions on x by nonnegativity restrictions) after dropping the redundant constraints (5.6) and (5.7). LP3 can be decomposed, using the Dantzig-Wolfe principle, by noting that the constraints (5.1) to (5.4) define a network flow problem which is closely related to n-paths. Let P' be the polyhedron formed by these constraints (5.1) to (5.4) and the nonnegativity restriction on x.

Lemma 4. All of the extreme points of P' are integers and are in one to one correspondence with the n-paths.

Proof: It is easily verified that each column of the constraint matrix defining P' contains exactly one + 1 and one -1. Therefore this matrix is totally unimodular and defines the possible shippings of one unit of flow through a related time expanded network [31].

As indicated in reference 11 (p. 130), such a problem is equivalent to a shortest path problem, and by (5.2) to (5.4) is actually an n-path problem in G. Furthermore any extreme point of P' is 0-1 valued and defines an n-path in G. Conversely, an n-path defines a feasible point in P', which is 0-1 valued, and must then be an extreme point. ∎

It appears that LP3 is equivalent to a network flow formulation of the TSP given by Roy [33, p. 549-50]. The complicating vertex constraints (5.5) are handled by Roy by adding additional arcs and introducing "bundle" constraints, that is generalized capacities for the sum of flows in sets of arcs. While Roy shows that this problem may be handled by Matthy's method [27], it is not yet proved that this method is tractable. However, integrality of the solution is not guaranteed and the following theorem shows that solving LP3 is equivalent to solving LP1.

Theorem 5. The optimal value of LP3 is z_{LP1}.

Proof: Applying the Dantzig-Wolfe decomposition principle to LP3, and using the previous lemma, provides the following master problem:

$$\text{Min} \quad \sum_{c \in C} d_c \lambda_c$$

$$\sum_{c \in C} \lambda_c = 1$$

$$\sum_{c \in C} a_{ci} \lambda_c = 1 \qquad i = 2,3,\ldots,n$$

$$\lambda_c \geq 0$$

In this problem the first constraint is just the convex combination constraint and the constraints 2 to n come from (5.5). As it was already observed in the proof of lemma 2 the first constraint is redundant, and dropping it results in LP1. ∎

In the following section we describe a subgradient optimization technique to solve LP2.

6. SUBGRADIENT OPTIMIZATION

Preliminary results indicate that the column generation scheme for solving LP1 is slow in converging to the optimal solution. Similar experience is supported by Held and Karp [16,17] in the context of the 1-tree relaxation for the symmetric TSP.

Since a branch and bound algorithm is to be employed, a lower bound on z_{LP1} could be used instead of the optimal value. According to a result by Dantzig et al. [7] a lower bound $\underline{z}$ for z_{LP1} at any given simplex iteration is given by

$$\underline{z} = z + \min_{c \in C} (d_c - \pi a_c)$$

where z is the current value of LP1 and π the corresponding dual variables. In general the behaviour of $\underline{z}$ is erratic and is not very good.

It is interesting to note that for a given set of vertex penalties π, a lower bound on the length of the optimal tour can be obtained by finding an n-path of minimum length. Held and Karp [17] developed a method called subgradient optimization (SGO) to approach optimal vertex penalties in the context of the symmetric TSP using 1-tree relaxations. Efficient implementation procedures and generalizations of SGO are discussed in references 6, 14, 18 and 19. The SGO method provides approximate optimal penalties in a reasonable number of iterations and the lower bounds generated using these penalties are satisfactory.

Let $\omega(\pi) = \min_{c \in C} (d_c - \pi a_c)$. Then the optimal penalty problem, LP2, can be stated as

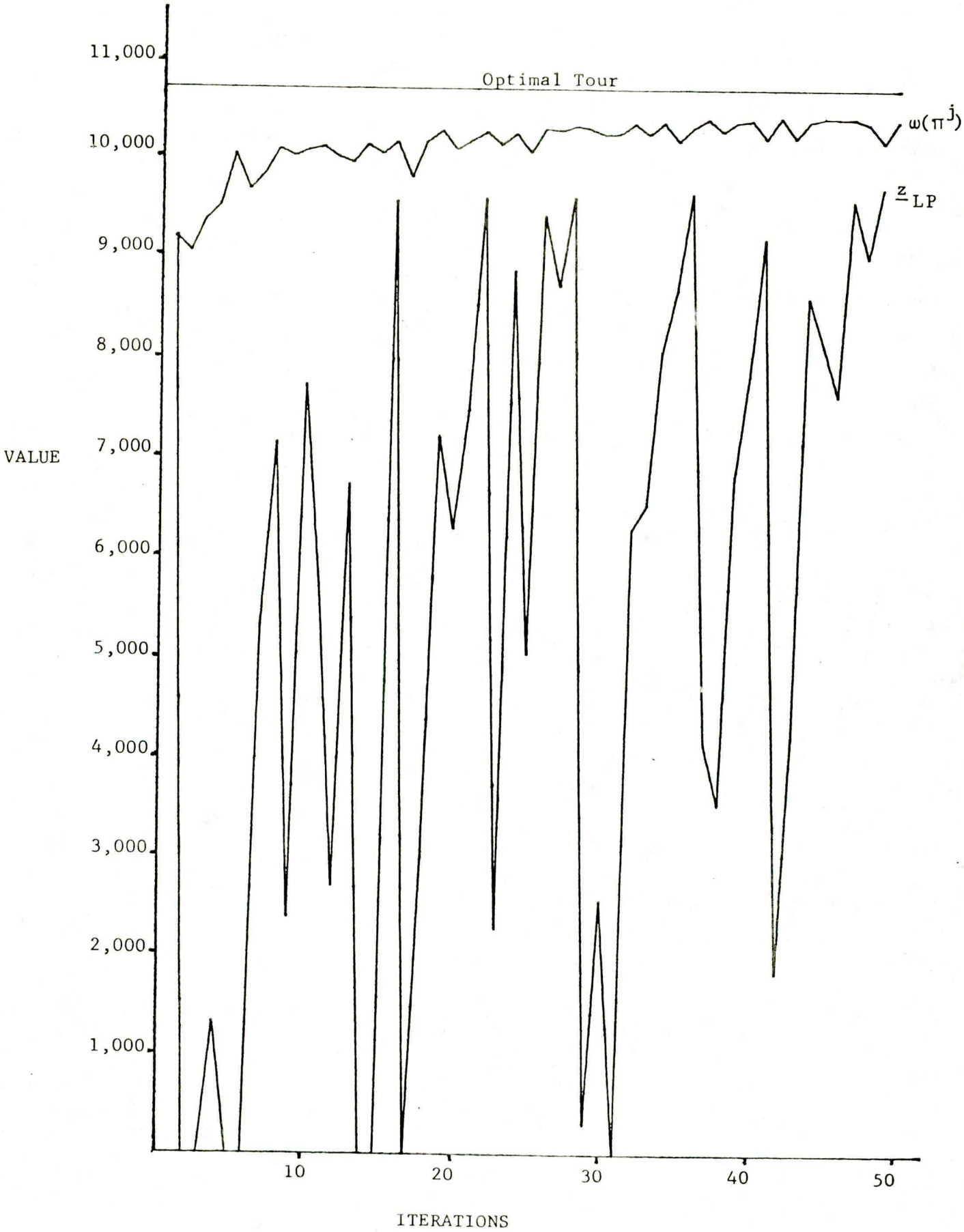
$$(OP) \quad \max \{\omega(\pi) \mid \pi \cdot \underline{1} = 0\}$$

Let a_π be the column vector associated with a shortest n-path c_π for the vertex penalties π. It is easily verified that $\underline{1} - a_\pi$ is a subgradient direction for ω, which may be used to define the iterative scheme $\pi^{j+1} = \pi^j + t_j(\underline{1} - a_{\pi j})$. (See references 6, 17, 19, and 39.)

We used $t_j = \lambda_j \cdot \frac{B - \omega(\pi^j)}{\|\underline{1} - a_{\pi j}\|^2}$ where λ_j is a sequence of real numbers and B is an upper bound on max $\omega(\pi)$. For example B can be set equal to the length of any tour provided by a heuristic algorithm. If $\lambda_j = 1$ the iteration can be viewed as an attempt to force $\omega(\pi^{j+1})$ to take the value B. For a discussion of the choice of t_j see reference 18.

It is necessary to define a stopping criterion as there is no guarantee that the iterative scheme terminates in a finite number of steps. If for some index j, $c_{\pi j}$ is a tour, then TSP is solved. Otherwise we stop the iterative process when the maximum value of $\omega(\pi)$ found so far has not improved for p iterations where p is parameter chosen in advance.

For a problem with 32 vertices, taken from reference 19, the simplex lower bound $\underline{z}_{LP}$ and the bound obtained using the subgradient algorithm are plotted as a function of the number of iterations. As can be seen from Figure 1, the behaviour of $\underline{z}_{LP}$ is erratic and the subgradient method provides better bounds.


11,000
10,000
9,000
8,000
7,000
6,000
5,000
4,000
3,000
2,000
1,000
VALUE
Optimal Tour
$\omega(\pi^j)$
$\underline{z}_{LP}$
10
20
30
40
50
ITERATIONS

7. BRANCH AND BOUND ALGORITHMS.

In this section we present two branch and bound algorithms for solving TSP, both of which incorporate the minimum n-path problem as a subproblem.

ALGORITHM 1.

The first algorithm is a standard branch and bound algorithm [12], with a LIFO problem selection rule. A fixed number of subgradient iterations are used in trying to fathom each subproblem. If fathoming does not occur then the subproblem is partitioned into two or more new subproblems. The branching rule will be illustrated by an example.

Branching Rule: Consider a problem with 6 vertices. Using a set of vertex penalties $\pi_2,\ldots,\pi_6$, suppose the shortest 6-path is c= {(1,2),(2,3),(3,2), (2,3), (3,6), (6,1)}. In the branching let F_1 denote the set of arcs in a subproblem which must be contained in an n-path and let F_2 be the set of excluded arcs. Then the following conditions partition the set of all tours.

(a) $F_1 = \emptyset$, $F_2 = \{(1,2)\}$

(b) $F_1 = \{(1,2)\}$ $F_2 = \{(2,3)\}$

(c) $F_1 = \{(1,2),(2,3)\}$ $F_2 = \emptyset$

Forcing an arc into a solution can be accomplished by joining the two corresponding vertices and thus reducing the graph of interest. For example, $F_1 = \{(1,2)\}$ joins vertices 1 and 2 eliminating arcs (1,3), (1,4), (1,5), (1,6), (3,2), (4,2), (5,2) and (6,2). Thus, in case (b), we want the shortest 5-path from vertex 2 to vertex 1 without arc (2,3). Arcs in F_2 are simply deleted from the graph or given infinite cost.

Suppose we solved case (b) and obtained the 5-path $c = \{(2,4), (4,5), (5,4), (4,5), (5,1)\}$. The subproblems generated here would be

(d) $F_1 = \{(1,2)\}$ $\quad F_2 = \{(2,3), (2,4)\}$

(e) $F_1 = \{(1,2), (2,4)\}$ $\quad F_2 = \{(4,5)\}$

(f) $F_1 = \{(1,2), (2,4), (4,5)\}$ $\quad F_2 = \emptyset$.

ALGORITHM 2.

The following algorithm is a specialization of an algorithm designed for the more general time-dependent traveling salesman problem, and the reader is referred to [31] for a detailed description. The algorithm also follows the lines of the general scheme sketched in [10, p. 77] and only the main difference with the previous algorithm are outlined below:

a) branching rule : A problem with origin 1 and intermediate vertices 2,...,n is separated into (n-1) subproblems according to the different choices of the vertex to be fixed in the last position. After a number of separations, a general subproblem is defined by its last k fixed vertices $i_k, i_{k-1}, \ldots, i_1$ forming a fixed path $i_k - i_{k-1} - \ldots - i_1 - 1$ with length denoted by FL. Such a subproblem may be considered as a TSP with origin 1 and (n-1)-k intermediate vertices, by replacing the length of the "last" arcs by d_{ji_k}.

b) dominance test : If there is some permutation $(j_k, j_{k-1}, \ldots, j_1)$ of the last k fixed vertices such that the resulting path $i-j_k-j_{k-1}-\ldots-j_1-1$ is shorter than $i-i_k-i_{k-1}-\ldots-i_1-1$ then the subproblem defined by fixing i in last position in the current subproblem cannot contain an optimum tour. While the problem of finding a shortest hamiltonian path with origin i and end 1, in the subgraph generated by the vertices $i, i_k, i_{k-1}, \ldots, i_1, 1$ is as difficult as the TSP itself, the dominance test checks the given path for an

improvement by trying adjacent pairwise interchanges and segment insertions, in the spirit of the work of Reiter and Sherman [32] (note that this method is distinct from the method used by Ignall and Schrague [20] for a similar dominance test.) This test is performed for each subproblem (with at least two fixed vertices) and each non-fixed vertex i, before the shortest path iterations. As soon as an improvement is found, the corresponding vertex is deleted from a list of "last position vertices" for the current subproblem.

c) Implicit bounds: For each penalty vector π, the dynamic programming algorithm used for the shortest path calculations, computes the values $b_j(\pi) = f(j,n-1) + d_{j1}$ (the minimum of which defines $f(1,n)$) with the modified distances, for the vertices j remaining in the last position list. Noting that $b_j(\pi)$ + FL (fixed length) is a valid lower bound for the best tour with j in last position, the maximum value B_j among those computed for the successive penalty vectors π may be used for fathoming (deleting j from the last position list when B_j is not less than the length of the imcumbent) and "redirecting" the subgradient.

d) Redirected subgradient: In place of taking the subgradient direction $\underline{1} - a_\pi$ associated with a shortest n-path, the redirected subgradient direction (which is called "modified subgradient" in [31]) is defined by the direction $\underline{1} - a_p$ associated with a shortest n-path with, as last vertex, a vertex j with least current implicit bound B_j. This avoids exploring unfruitful regions of the penalty space. (A related idea is called a "resource-space tour" in [26]).

e) Exploration strategy: When a given number of subgradient iterations have have been performed in the current subproblem, it is branched from, in an attempt to refine the bounds, according to the remaining vertices in the

last position list. The corresponding subproblems are added to the list of problems in an order of non-increasing values of their implicit bounds B_j, and the last subproblem in the list is then selected (LIFO rule). When the shortest n-path (defining the redirected subgradient) is a tour, or when the last position list becomes empty, the current problem is deleted from the list, and the last subproblem remaining in the list is selected (backtracking) until this list is empty.
For more details, see reference 31.

A run of the above algorithm is defined by the distance matrix and by a set of user parameters p_1 - p_4. The parameter p_1(resp. p_2) is the number of shortest path iterations after which the initial problem (resp. a subproblem) is branched from. When no improvement in the least implicit bound B_j is found within p_3 successive iterations, the step size (which is initially 1) in (6-1) is multiplied by p_4 (step size reduction rate, $0 < p_3 \leq 1$). See [13] for a discussion of the convergence of the subgradient iterations with respect to the values of p_3 and p_4. An alternate strategy is the "modified subgradient" described in [6] (see also [24], [39]). However, the use of the redirected subgradient may invalid the theoretical results or empirical observations about the various subgradient strategies.

8. REFINEMENTS

A Tighter Relaxation of TSP.

Consider any n-path $c = \{(1,i_1), (i_1,i_2),\ldots(i_{n-1},1)\}$. It has been observed that for many n-paths, $i_k = i_{k+2}$ for several values of k. An n-path c is said to contain an oscillation if there exists a k such that $i_k = i_{k+2}$ for some k. Since a tour cannot contain an oscillation, we are interested only in n-paths without oscillations. For an n vertex problem the number of n-paths from vertex 1 to vertex 1 is $(n-1)\ (n-2)^{n-2}$. The number of n-paths without oscillations is $(n-1)\ (n-2)\ (n-3)^{n-3}$. The ratio of the number of n-paths to the number of n-paths without oscillations is

$$\left(\frac{n-2}{n-3}\right)^{n-3} = \left(1+\frac{1}{n-3}\right)^{n-3}$$

which is approximately e(=2.718...) for sufficiently large n. Similarly, it is easy to prove that the ratio of the number of n-paths to the number of n-paths without k-oscillations (i.e. n-paths in which no vertex appears more than once in any sequence of k consecutive vertices.) is asymtotically e^{k-1}. Thus an advantage of restricting attention to n-paths without oscillations is that it provides a better lower bound for TSP. We describe below an algorithm for finding an n-path of minimum length which does not contain oscillations.

1. Initialization(t= 2)

Let I denote the set of integers 2,3,...,n. For each $i \in I$ let

$$f(i,2) = \min_{j \in I}\{d_{1j} + d_{ji}\}$$

$$= d_{1j^*} + d_{j^*i},$$

$$m(i,2) = j^*,$$

$$f'(i,2) = \min_{k \in I}\{d_{1k} + d_{ki} \mid k \neq j^*\}$$

$$= d_{1k^*} + d_{k^*i}$$

and $$m'(i,2) = k^*.$$

2. General Step ($t = 3,\ldots,n-1$)

For t running from 3 to $n-1$ do the following.

Define the auxiliary table F for $i \in I$ and $j \in I$ as

$$F(i,j) = \begin{cases} f'(j,t-1) + d_{ji} & \text{if } i = m(j,t-1) \\ f(j,t-1) + d_{ji} & \text{otherwise} \end{cases}$$

for each $i \in I$, let

$$f(i,t) = \min_{j \in I} \{F(i,j)\}$$

$$= F(i,j^*),$$

$$m(i,t) = j^*,$$

$$f'(i,t) = \min_{k \in I} \{F(i,k) \mid k \neq m(i,t)\}$$

$$= F(i,k^*),$$

and $m'(i,t) = k^*$.

3. Last Step. ($t = n$).

The length of the n-path of minimum length which does not contain oscillations is given by

$$d_c = \operatorname{Min}_{i \in I} \{f(i,n-1) + d_{i1}\}$$

$$= f(i^*,n-1) + d_{i^*1} \quad \text{for some } i^* \in I.$$

The n-path of minimum length can be retrieved using the following iterative scheme.

$$i_{n-1} = i^* \qquad \text{(see step 3)}$$

$$i_{n-2} = m(i_{n-1},n-1)$$

For $t = n-3,\ldots,1$, let i_t be given by

$$i_t = \begin{cases} m(i_{t+1}, t+1) & \text{if } m(i_{t+1}, t+1) \neq i_{t+2} \\ m'(i_{t+1}, t+1) & \text{otherwise.} \end{cases}$$

Then $\{(1,i_1), (i_1,i_2),\ldots,(i_{n-1},1)\}$ is the required n-path of minimum length.

The following lemma provides the justification of the above procedure.

Lemma 6: For $t=2,\ldots,n-1$, $f(i,t)$ is the length of a minimum t-path without oscillation from vertex 1 to vertex i and $f'(i,t)$ is the minimum length of such a t-path with $i'_{t-1} \neq m(i,t)$.

Proof: The result is true for $t = 2$ and all $i \in I$. Assume that it is true for all $i \in I$ and $t \leq s$. By induction, when $t = s+1$, the algorithm defines $F(i,j)$ to be the length of the shortest $s+1$-path from vertex 1 to vertex j without oscillation which has arc (i,j) as the last arc. Thus $f(j,s+1)$ and $f'(j,s+1)$ are defined correctly for each $j \in I$. ∎

It can be verified that the above algorithm requires $O(n^3)$ additions and $O(n^3+n^2 \log_2 n)$ comparisons. Thus one can see that the additional effort required by this algorithm is minimal when compared with the algorithm of section 2.

9. COMPUTATIONAL EXPERIENCE

The computational results reported in this section are divided into two parts; a study of the bounds generated by the n-path relaxation and results from the implementation of the two algorithms discussed in section 7.

To compare the bounds generated by different methods complete graphs were generated with random costs ranging from 0-10 up to 0-200. Both symmetric and asymmetric graphs were generated and each problem was solved both with the original cost matrix as well as with the shortest distance matrix (which satisfies the triangle inequality). The 1-tree bounds were found for the symmetric problems and the 1-arborescence bounds (mentioned by Held and Karp [16]) were found for the asymmetric problems. The n-path bounds with and without oscillations were found for all problems. Column generation was used to find these bounds for each method.

Table 1 summarizes the results found. The three entries under each method are the average bound obtained, the average number of simplex iterations needed, and the number of times a tour was generated.

TABLE 1

No. of Cities	No. of Problems	Symmetric	Triangle Inequality	n-Path	n-Path Without Oscillations	1-Tree or 1-Arborescence
10	10	No	Yes	62.9	63.9	63.9
				40.5	30.2	47.6
				3	5	2
10	10	No	No	66.6	68.1	67.8
				41.9	35.6	64.8
				4	6	4
10	10	Yes	Yes	148.7	160.7	162.1
				22.9	31.6	42.1
				0	5	2
10	10	Yes	No	166.5	184.7	184.7
				27.7	39.4	55.0
				0	9	9
15	10	No	Yes	51.2	51.3	50.8
				179.7	137.4	629.8
				2	2	0
15	10	No	No	53.5	54.0	54.3
				181.5	282.6	444.3
				2	3	5
15	10	Yes	Yes	177.4	197.7	203.0
				60.2	93.4	188.6
				0	1	0
15	10	Yes	No	197.3	236.4	236.4
				73.7	443.5	350.2
				0	5	5

It should be noted that n-paths without oscillations did significantly better than n-paths with oscillations in the symmetric graph case. Also n-paths without oscillations did about the same as the 1-trees and the 1-arborescences, while in general taking fewer iterations. In addition the n-paths without oscillations yielded tours in 36 of the problems to only 27 for the 1-trees and 1-arborescences.

These results make it clear that minimal 1-trees are still better for symmetric problems as they can be solved in $O(n^2)$ time. (It should be noted that the time required for one shortest path iteration for symmetric problems can be cut in half by using $d_{ij} - \frac{1}{2}\pi_i - \frac{1}{2}\pi_j$.) Although Tarjan [38] has recently presented on $O(n^2)$algorithm for finding minimal 1-arborescences it is not yet clear whether they are better, due to the increased number of iterations. Table 2 gives the somewhat disappointing results of implementing algorithms 1 and 2. Again random graphs were generated and n-paths without oscillations were used as the subproblems.

TABLE 2

Algorithm	No. of Cities	No. of Problems	Average No. Of Shortest Paths Found	Average Execution Time*(sec)
1	15	10	61	9.8
1	20	5	105	36.2
2	20	3	127	9.2
2	30	3	-	95.3
2	40	1	-	>240

Although these results do not show good running time the number of branches selected in the enumeration trees were in general quite small (less than 10 in most cases). Thus if the n-paths computation could be reduced, or something like the "cell cost operators" of Srinivasan and Thompson [37] developed,

*Algorithm 1 was run on a UNIVAC 1108 and algorithm 2 on a CDC Cyber 74-18.

this method would show considerable promise.

One area where this method does work well is in problems with special structure in the cost matrix. Reference 31 shows excellent results for the time dependent traveling salesman problems.

REFERENCES

1. A. Aho, J. Hopcroft, and J. Ullman, The Design and Analysis of Computer Algorithms, Addison-Wesley (1974).

2. M.S. Bazaraa and J.J. Goode, "The Traveling Salesman Problem: A Duality Approach," Math. Prog. 13, 221-237 (1977)

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

4. M. Bellmore and J.C. Malone, "Pathology of Traveling Salesman Subtour Elimination Algorithms", Operations Research 19, 278-307 (1971).

5. P.M. Camerini, L. Fratta and F. Maffioli, "A Heuristically Guided Algorithm for the TSP," Journal of the Institution of Computer Science 4, 2, 31-35 (1973).

6. P.M. Camerini, L. Fratta and F. Maffioli, "On Improving Relaxation Methods by Modified Gradient Techniques," Math. Prog. Study 3, p. 26-34 (1975).

7. G.B. Dantzig, D.R. Fulkerson, and S.M. Johnson, "Solution of a Large Scale Traveling Salesman Problem," Operations Research 2, 393-410 (1967).

8. J. Edmonds, "Paths, Trees and Flowers", Canadian J. Math. 17, 339-467 (1965).

9. J. Edmonds, "Optimal Branchings", J. Research, National Bureau of Standards 71B, 233-240 (1967)

10. M.L. Fisher, W.D. Northup and J.F. Shapiro, "Using Duality to Solve Discrete Optimization Problems: Theory and Computational Experience", Math. Prog. Study 3, p. 56-94 (1975).

11. L.R. Ford and D.R. Fulkerson, Flows in Networks, Princeton University Press (1962).

12. R.S. Garfinkel and G.L. Nemhauser, Integer Programming, John Wiley (1972).

13. J.L. Goffin, "On Convergence Rates of Subgradient Optimization Methods," Math. Prog. 13, 329-347 (1977).

14. H. Hansen and J. Krarup, "Improvements of the Held-Karp Algorithm for the Symmetric Traveling Salesman Problem," Math. Prog. 7, 87-96 (1974).

15. G. Hadley, Nonlinear and Dynamic Programming, Addison-Wesley (1964)

16. M. Held and R.M. Karp, "The Traveling Salesman Problem and Minimum Spanning Trees," Operations Research 18, 1138-1162 (1970).

17. M. Held and R.M. Karp, "The Traveling Salesman Problem and Minimum Spanning Trees: Part II," Math. Prog. 1, 6-25 (1971)

18. M. Held, R.M. Karp, and P. Wolfe, "Large Scale Optimization and the Relaxation Method", Proc. ACM, 507-509 (1972).

19. M. Held, P. Wolfe, and H. Crowder, "Validation of Subgradient Optimization," Math. Prog. 6, 62-68 (1974).

20. E. Ignall and L.E. Schrague, "Application of the Branch and Bound to Some Flow Shop Scheduling Problems," Operations Research, 13, 3, (1965).

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

22. R.M. Karp, "Reducibility Among Combinatorial Problems," in R. Miller and J. Thatcher (eds), Complexity of Computer Computations, Plenum Press (1972).

23. L.S. Lasdon, Optimization Theory for Large Systems, Macmillan (1970).

24. C. Lemarechal, "An Extention of Davidon Methods to Non Differentiable Problems," Math. Prog. Study 3, p. 95-109, (1975)

25. J.D. Little, K.G. Murty, D.W. Sweeny, and C. Kaul, "An Algorithm for the Traveling Salesman Problem, "Operations Research 11, 974-989 (1963)

26. R.E. Marsten and T.L. Morin, "A Hybrid Approach to Discrete Mathematical Programming, " Working Paper OR 051-76, Operations Research Center, MIT, (1976).

27. G. Matthys, "Flot Optimum dans un Réseau à Capacités de Faisceaux," Actes du 2ième Congrès International de Recherche Opérationnelle 164-170, Dunod, Paris, (1961).

28. P. Miliotis, "Integer Programming Approaches to the Traveling Salesman Problem," Math. Prog. 10, 367-378 (1976).

29. G.L. Nemhauser, Dynamic Programming, John Wiley (1966).

30. M. Padberg and M.R. Rao, "The Traveling Salesman Problem and a Class of Polyhedra of Diameter Two," Math. Prog. 7, 32-45, (1974).

31. J.C. Picard and M. Queyranne, "The Time-Dependent Traveling Salesman Problem and Application to the Tardiness Problem in One-Machine Scheduling," Operations Research 26, 86-110 (1978).

32. S. Reiter and G. Sherman, "Discrete Optimizing," Journal SIAM 13, 3 864-889 (1965).

33. B. Roy, Algèbre Moderne et Théorie des Graphes, 2, 549-50 Dunod, Paris (1970).

34. R. Saigal, "A Constrained Shortest Route Problem", Operations Research 16, 388-401 (1970).

35. D.M. Shapiro, "Algorithms For the Solution of the Optimal Cost and Bottleneck Traveling Salesman Problems," Sc.D. Thesis, Washington University, St. Louis (1966).

36. T.H.C. Smith, V. Srinivasan, and G.L. Thompson, "Computational Performance of Three Subtour Elimination Algorithms for Solving the Asymmetric Traveling Salesman Problem," Report no. 369, Carnegie-Mellon Univ. (1975).

37. V. Srinivasan and G.L. Thompson, "Solving Scheduling Problems by Applying Cost Operators to Assignment Models", in Symposim on the Theory of Scheduling and its Applications,S.E. Elmaghraby (Ed.), Springer-Verlag, N.Y. p. 399-425, (1973).

38. R.E. Tarjan, "Finding Optimal Branchings", Networks 7, 25-35 (1977).

39. P. Wolfe, "A Method of Conjugate Subgradients for Minimizing Non Differentiable Functions", Math. Prog. Study 3, p. 145-173, (1975).