THE TRAVELING-SALESMAN PROBLEM AND MINIMUM SPANNING TREES

Michael Held

IBM Systems Research Institute, New York, New York

and

Richard M. Karp

University of California, Berkeley, California

(Received September 2, 1969)

This paper explores new approaches to the symmetric traveling-salesman problem in which 1-trees, which are a slight variant of spanning trees, play an essential role. A 1-tree is a tree together with an additional vertex connected to the tree by two edges. We observe that (i) a tour is precisely a 1-tree in which each vertex has degree 2, (ii) a minimum 1-tree is easy to compute, and (iii) the transformation on 'intercity distances' $c_{ij} \rightarrow c_{ij} + \pi_i + \pi_j$ leaves the traveling-salesman problem invariant but changes the minimum 1-tree. Using these observations, we define an infinite family of lower bounds $w(\pi)$ on C^*, the cost of an optimum tour. We show that $\max_\pi w(\pi) = C^*$ precisely when a certain well-known linear program has an optimal solution in integers. We give a column-generation method and an ascent method for computing $\max_\pi w(\pi)$, and construct a branch-and-bound method in which the lower bounds $w(\pi)$ control the search for an optimum tour.

THIS PAPER explores new approaches to the symmetric traveling-salesman problem in which 1-trees, which are a slight variant of spanning trees, play an essential role. A 1-tree is a graph with vertices $1, 2, \cdots, n$ consisting of a tree on the vertices $2, 3, \cdots, n$ together with two edges incident with vertex 1. We observe that (i) a tour is precisely a 1-tree in which each vertex has degree 2, (ii) a minimum 1-tree is easy to compute, and (iii) the transformation on 'intercity distances' $c_{ij} \rightarrow c_{ij} + \pi_i + \pi_j$ leaves the traveling-salesman problem invariant but changes the minimum 1-tree.

We consider the systematic variation of the π_i to produce a situation in which a minimum 1-tree is a tour. However, we show by example that this is not always possible; i.e., that there may be an unavoidable 'gap' between the two problems. We define a nonnegative function $f(\pi)$ that measures the size of this gap, and assumes the value 0 at its minimum precisely when it is possible to obtain a minimum 1-tree that is a tour.

We are thus led to the problem of minimizing $f(\pi)$. We give two methods of doing so. The first expresses the problem as a linear program that can be solved by a column-generation technique. This linear program is

related to a well-known integer linear programming formulation for the traveling-salesman problem, and it is shown that the gap phenomenon corresponds to the possible existence of nonintegral extreme points. An ascent method for minimizing the gap is also specified.

Finally, the ascent method is embedded within an implicit enumeration procedure that, in principle, always produces an optimum tour.

We also indicate some further applications, develop relations with matroids, and cite some limited computation experience.

1. THE TRAVELING-SALESMAN PROBLEM AND A RELATED SPANNING-TREE PROBLEM

LET K_n BE the complete undirected graph on the vertex set $\{1, 2, \cdots, n\}$. The edge incident with vertices i and j is denoted $\{i, j\}$, and is assigned

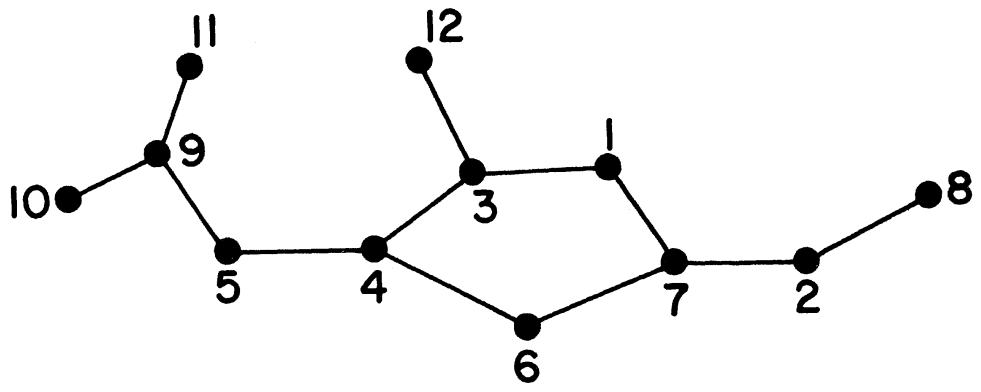


Fig. 1. A 1-tree.

the *weight* c_{ij}. The weight of a subgraph is the sum of the weights of its edges. The *traveling-salesman problem* seeks a *tour* (i.e., a cycle passing through each vertex exactly once) of minimum weight (reference 1 is a recent survey of work on this problem). A connected graph without cycles is called a *tree*. The familiar *minimum spanning-tree problem* seeks a tree of minimum weight on the vertex set $\{1, 2, \cdots, n\}$. For our purposes, a variant of this problem is required, in which we seek a minimum-weight 1-*tree*. A 1-tree consists of a tree on the vertex set $\{2, 3, \cdots, n\}$, together with two distinct edges at vertex 1 (cf. Fig. 1). Thus, a 1-tree has a single cycle, this cycle contains vertex 1, and vertex 1 always has degree two. Whereas the traveling-salesman problem is considered very difficult, the minimum spanning-tree problem can be solved by an algorithm that inspects each edge of K_n exactly once[4,12] (*see also* section 6). A minimum-weight 1-tree can be found by constructing a minimum spanning tree on the vertex set $\{2, \cdots, n\}$, and then adjoining two edges of lowest weight at vertex 1.

Every tour is a 1-tree, and a 1-tree is a tour if and only if each of its vertices has degree 2. Clearly, if a minimum-weight 1-tree is a tour, it is the solution to the traveling-salesman problem.

LEMMA 1. *Let* $\pi = (\pi_1, \pi_2, \cdots, \pi_n)$ *be a real* n*-vector. If* C^* *is a minimum-weight tour with respect to the edge weights* c_{ij}*, then it is also a minimum-weight tour with respect to the edge weights* $c_{ij}+\pi_i+\pi_j$.

Proof. The weight of a tour C with respect to the weights c_{ij} is

$$\sum_{\{i,j\}\in C} c_{ij},$$

and with respect to $c_{ij}+\pi_i+\pi_j$, is $\sum_{\{i,j\}\in C} (c_{ij}+\pi_i+\pi_j)$. Since each vertex is incident with two edges of C, the second summation minus the first is exactly $2\sum_{i=1}^{i=n} \pi_i$. Hence, in changing from one set of weights to the other, the costs of all tours are changed by the same amount, and the comparison between tours is unaffected.

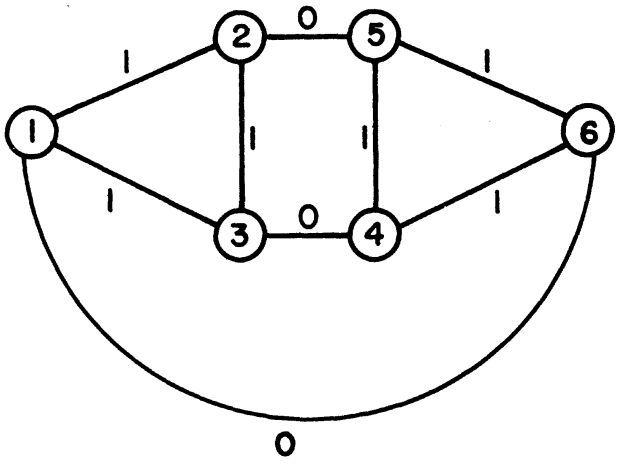


Figure 2.

We remark that the transformation from edge weights c_{ij} to $c_{ij}+\pi_i+\pi_j$ does, in general, affect the identity of a minimum spanning 1-tree. Thus, a possible strategy for solving the traveling-salesman problem is to seek a vector π such that a minimum 1-tree with respect to weights $c_{ij}+\pi_i+\pi_j$ is a tour. To this end, we introduce a 'gap' function $f(\pi)$ equal to the amount by which the cost of a minimum tour with respect to the weights $c_{ij}+\pi_i+\pi_j$ exceeds the cost of a minimum 1-tree with respect to the same weights. Then $f(\pi)$ is a nonnegative function, and it assumes the value zero at its global minimum, if at all. An example given at the end of this section shows that $f(\pi)$ can, in general, be positive at its minimum point. However, even in this case it is useful in connection with implicit search procedures (cf. section 5) to find a π^* such that $f(\pi^*)$ is positive but small, since an optimum tour with respect to the weights $c_{ij}+\pi_i^*+\pi_j^*$ is then a near-minimum 1-tree. Thus, we consider the problem of finding $\min_\pi f(\pi)$.

We now show that this problem can be expressed as a linear program. Let the 1-trees of K_n be indexed $1, 2, \cdots, q$, with k as a generic index. By definition,

$$f(\pi) = W + 2\sum_{i=1}^{i=n} \pi_i - \min_k [c_k + \sum_{i=1}^{i=n} \pi_i d_{ik}],$$

where W is the weight of a minimum tour with respect to the weights c_{ij}, c_k is the weight of the kth 1-tree with respect to the weights c_{ij}, and d_{ik} is the degree of vertex i in the kth 1-tree. Thus,

$$f(\pi) = W - \min_k [c_k + \sum_{i=1}^{i=n} \pi_i (d_{ik} - 2)],$$

and determining $\min_\pi f(\pi)$ is equivalent to finding $\max_\pi w(\pi)$, where

$$w(\pi) = \min_k [c_k + \sum_{i=1}^{i=n} \pi_i v_{ik}],$$

where we have denoted $d_{ik} - 2$ by v_{ik}. This problem is equivalent to the following linear programming problem:

$$\max w \text{ subject to: } w \leqq c_k + \sum_{i=1}^{i=n} \pi_i v_{ik}. \qquad (k = 1, 2, \cdots, q) \quad (1)$$

Dualizing, we obtain:

$$\min \sum_k c_k y_k; \quad y_k \geqq 0, \quad \sum_k y_k = 1, \quad \sum (-v_{ik}) y_k = 0. \quad (i = 2, 3, \cdots, n-1). \qquad (2)$$

This program seeks a minimum-weight 'convex combination of 1-trees' such that each vertex has, on the average, degree two.

We conclude this section with an example showing that the minimum of the gap $f(\pi)$ may not be zero.

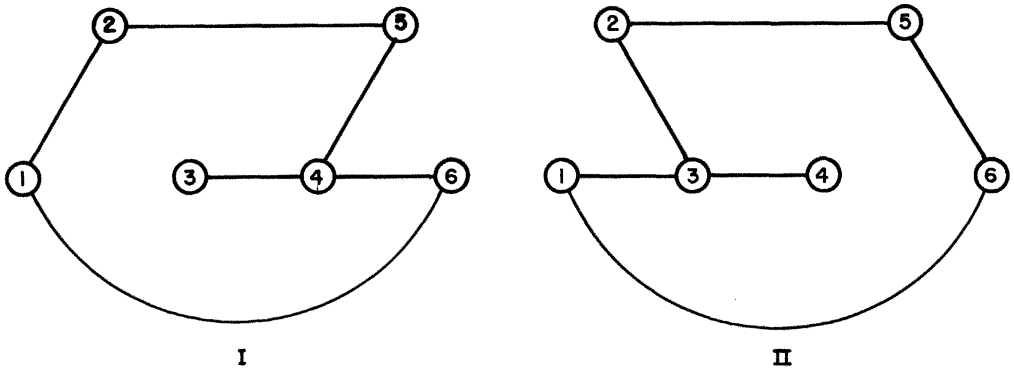
Figure 2 gives the weights c_{ij} for the vertex set $\{1, 2, \cdots, 6\}$; edges not explicitly shown in the figure are assumed to have a very large positive weight. Each of the two 1-trees shown in Fig. 3 is a minimum 1-tree with weight 3; the degree for each vertex in each 1-tree is also shown. If each 1-tree is assigned a coefficient y_k of $\frac{1}{2}$, it is clear that we have a solution of program (2), and the value of the objective function is 3. However, a solution to the traveling-salesman problem is given in Fig. 4, and has weight 4. Hence, in this case, the minimum value of $f(\pi) = 4 - 3 = 1$.

2. RELATION TO A LINEAR PROGRAM

IT IS WELL known[2,3,14] that the optimal tours for the traveling-salesman problem with weights c_{ij} correspond to optimal solutions of a certain integer linear program. In this section, we show that minimizing the gap $f(\pi)$ is equivalent to solving this program *without* the integer constraints. The integer linear program representing the traveling-salesman problem is as follows:

$$\min \sum_{1 \leqq i < j \leqq n} c_{ij} x_{ij}$$

subject to



DEGREES

Vertex	1-Tree I	1-Tree II
2	2	2
3	1	3
4	3	1
5	2	2
6	2	2
y_k	1/2	1/2

Figure 3.

$$
\begin{aligned}
&\text{(i)}\quad \textstyle\sum_{j>i} x_{ij}+\sum_{j<i} x_{ji}=2, && (i=1, 2, \cdots, n)\\
&\text{(ii)}\quad \textstyle\sum_{\substack{i\in S\\ j\notin S\\ i<j}} x_{ij}\geqq |S|-1, \quad \text{for any proper subset } S\subset\{2, 3, \cdots, n\}, && (3)\\
&\text{(iii)}\quad 0\leqq x_{ij}\leqq 1, && (1\leqq i<j\leqq n)\\
&\text{(iv)}\quad x_{ij} \text{ integer.}
\end{aligned}
$$

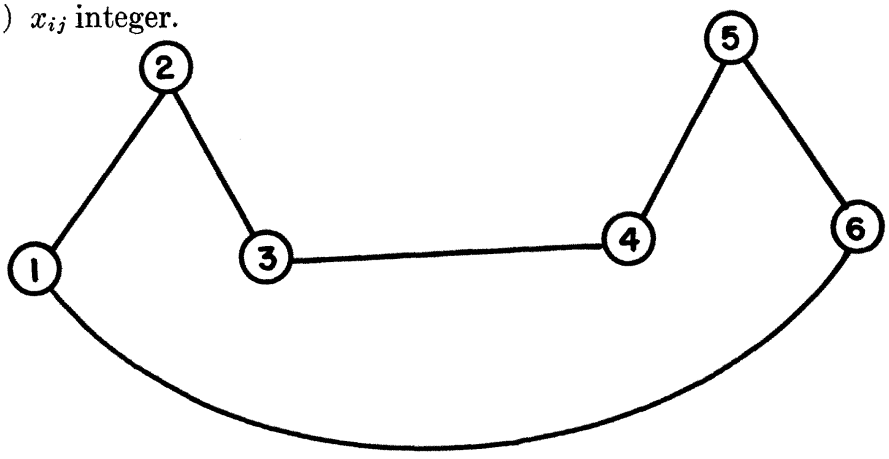


Figure 4.

The feasible solutions of this program are in one-to-one correspondence with the tours. The tour corresponding to a feasible solution (x_{ij}) includes edge $\{i, j\}$ if and only if $x_{ij}=1$. The constraints (i) ensure that each vertex has degree 2, and the constraints (ii) ensure that no cycle is formed among a proper subset of the vertices.

LEMMA 2. *In program* (3), *the constraints* (i) *may be replaced by*

$$\sum_{j>i} x_{ij}+\sum_{j<i} x_{ji}=2, \quad i=1, 2, \cdots, n-1, \quad \text{and} \quad \sum_{1\leqq i<j\leqq n} x_{ij}=n.$$

Proof.

$$\begin{aligned}\sum_{1\leqq i<j\leqq n} x_{ij} &= (\tfrac{1}{2}) \sum_{i=1}^{i=n} \Big(\sum_{j>i} x_{ij}+\sum_{j<i} x_{ij}\Big) \\ &= (\tfrac{1}{2}) \sum_{i=1}^{n-1} \Big(\sum_{j>i} x_{ij}+\sum_{j<i} x_{ji}\Big)+(\tfrac{1}{2}) \sum_{j} x_{jn}.\end{aligned}$$

Thus

$$\sum_{1\leqq i<j\leqq n} x_{ij}=(\tfrac{1}{2}) \sum_{i=1}^{n-1} 2+(\tfrac{1}{2}) \sum_{j} x_{jn},$$

$$\sum_{1\leqq i<j\leqq n} x_{ij}=n-1+(\tfrac{1}{2}) \sum_{j} x_{jn}.$$

Clearly, then, in the presence of the other constraints, $\sum_{1\leqq i<j\leqq n} x_{ij}=n$ if and only if $\sum_{j} x_{jn}=2$.

Thus, we may rewrite program (3) in the following way:

$$\min \sum_{1\leqq i<j\leqq n} c_{ij}x_{ij}$$

subject to

$$\begin{aligned}&\text{(i)} && \sum_{j>i} x_{ij}+\sum_{j<i} x_{ji}=2, && (i=2, 3, \cdots, n-1)\\ &\text{(ii)} && \sum_{j} x_{1j}=2,\\ &\text{(iii)} && \sum_{1\leqq i<j\leqq n} x_{ij}=n,\\ &\text{(iv)} && \sum_{\substack{i\in S\\ j\notin S\\ i<j}} x_{ij}\geqq |S|-1, \text{ for any proper subset } S\subset\{2, 3, \cdots, n\}, && (3')\\ &\text{(v)} && x_{ij}\leqq 1,\\ &\text{(vi)} && x_{ij}\geqq 0,\\ &\text{(vii)} && x_{ij} \text{ integer.}\end{aligned}$$

In order to write program (3′) in a more compact form, we introduce matrix-vector notation. Let x and c be $\binom{n}{2}$-vectors with components x_{ij} and $c_{ij}(1\leqq i<j\leqq n)$, respectively. Denote the constraints (i) by $Ax=b$, and the constraints (ii), (iii), (iv), and (v) by $A'x\leqq b'$ where A and A' are appropriately defined matrices and b and b' are appropriately defined vectors [note that b is an $(n-2)$-vector, all of whose components are 2]. Program (3′) may thus be written as

$$\min_x \{cx|Ax=b, A'x\leqq b', x\geqq 0, x \text{ integer}\}.$$

The following theorem characterizes the extreme points of the polyhedron $A'x \leqq b'$, $x \geqq 0$ as the adjacency matrices of 1-trees.

THEOREM 1. *Let* $T^1, T^2, \cdots, T^k, \cdots, T^q$ *be the* 1-*trees on the vertex set* $\{1, 2, \cdots, n\}$. *Associate with* T^k *the* $\binom{n}{2}$-*vector* (e_{ij}^k) *defined by*:

$$(e_{ij}^k) = \begin{cases} 1 \text{ if } (i,j) \text{ is an edge of } T^k, \\ 0 \text{ otherwise.} \end{cases} \qquad (1 \leqq i < j \leqq n)$$

Then the extreme points of the polyhedron $A'x \leqq b'$, $x \geqq 0$ *are the points* (e_{ij}^k).

Clearly, the points (e_{ij}^k) belong to the polyhedron $A'x \leqq b'$, $x \geqq 0$: they satisfy constraint (ii) of program (3′) since vertex 1 always has degree 2 in a 1-tree; they satisfy constraint (iii) since every 1-tree has n edges; they satisfy constraints (iv) since the only cycle in a 1-tree contains vertex 1; they satisfy (v) trivially. That these points are the extreme points is a special case of a theorem stated in section 6.

As a consequence of Theorem 1, a linear program of the form

$$\min_x \{dx | A'x \leqq b', x \geqq 0\}$$

represents the problem of finding a minimum 1-tree with respect to the weights d_{ij}.

In particular, the linear program

$$\min_x \{cx + \tilde{\pi} A x | A'x \leqq b', x \geqq 0\},$$

where $\tilde{\pi}$ is the $(n-2)$ vector $(\pi_2, \pi_3, \cdots, \pi_{n-1})$, represents the problem of finding a minimum 1-tree with respect to the weights $c_{ij} + \pi_i + \pi_j$ with $\pi_1 = \pi_n = 0$. (It is possible to restrict attention to the case $\pi_1 = \pi_n = 0$ since $f(\pi)$ is independent of π_1, and depends only on the differences $\pi_i - \pi_n$, $i = 2, 3, \cdots, n-1$.)

Recalling that the gap $f(\pi)$ is the difference between the weight of a minimum tour with respect to $c_{ij} + \pi_i + \pi_j$ and the weight of a minimum 1-tree with respect to $c_{ij} + \pi_i + \pi_j$, and that b is a vector all of whose components are 2, we obtain

$$\begin{aligned} f(\pi) &= \min_x \{cx | Ax = b, A'x \leqq b', x \geqq 0, x \text{ integer}\} \\ &\quad + \tilde{\pi} b - \min_x \{cx + \tilde{\pi} A x | A'x \leqq b', x \geqq 0\} \\ &= \min_x \{cx | Ax = b, A'x \leqq b', x \geqq 0, x \text{ integer}\} \\ &\quad - \min_x \{cx + \tilde{\pi}(Ax - b) | A'x \leqq b', x \geqq 0\}, \end{aligned}$$

and

$$w(\pi) = \min_x \{cx + \tilde{\pi}(Ax - b) | A'x \leqq b', x \geqq 0\}.$$

We have seen that minimizing the gap $f(\pi)$ is equivalent to maximiz-

ing $w(\pi)$. The following theorem relates this maximization problem to program (3).

THEOREM 2. $\max_\pi w(\pi) = v$, *where*

$$v = \min_x \{cx | Ax = b,\ A'x \leqq b',\ x \geqq 0\}. \tag{4}$$

Proof. By definition,

$$v = \min_x \{cx | Ax = b,\ A'x \leqq b',\ x \geqq 0\}.$$

Dualizing, we obtain

$$\begin{aligned} v &= \max_{u,u'} \{-ub - u'b' | uA + u'A' \geqq -c,\ u' \geqq 0\}, \\ v &= \max_u [\max_{u'} \{-ub - u'b' | uA + u'A' \geqq -c,\ u' \geqq 0\}]. \end{aligned} \tag{5}$$

Dualizing the inner maximization problem, with u regarded as constant, we obtain

$$v = \max_u [\min_x \{cx + u(Ax - b) | A'x \leqq b',\ x \geqq 0\}].$$

Hence, setting $u = \pi$, $v = \max_\pi w(\pi)$.

COROLLARY 1. *The gap $f(\pi)$ assumes the value zero for some π if and only if the linear program* (4) [*which results from removing the integer constraints* (iv) *in* (3)], *has an optimal solution $\bar{x}$ such that $\bar{x}$ is integer.*

COROLLARY 2. *If (u, u') is an optimal solution of* (5), *then*

$$w(u) = \max_\pi w(\pi).$$

We note that, if the Dantzig-Wolfe decomposition principle is applied to (4) using the characterization of the extreme points of the polyhedron $A'x \leqq b'$, $x \geqq 0$ given in Theorem 1, the linear program (2) is obtained, and this provides an alternate proof of Theorem 2.

It is of interest that one of the earliest attacks on the traveling-salesman problem[2,3] took the solution of (4) as its starting point. In view of the development in the present section, it is equivalent to solve any of the programs (1), (2), (4), or (5), and doing so will solve the traveling-salesman problem precisely when the 'gap' $\min_\pi f(\pi)$ is zero. Among these programs, (2) seems the most promising, since it has the fewest constraints, and hence the smallest bases; this program is discussed more fully in Section 3.

3. A COLUMN-GENERATION TECHNIQUE

IN THIS SECTION we show that program (2) can be solved by a column-generation technique in which a minimum 1-tree calculation determines the column to enter the basis. Recall that in program (2) each variable y_k corresponds to a 1-tree T^k. The associated cost coefficient c_k is the weight

of T^k with respect to the c_{ij} and the associated column of the constraint matrix has the form

$$(1, -v_{2k}, -v_{3k}, \cdots, -v_{n-1,k})^T,$$

where $v_{ik}=d_{ik}-2$, and d_{ik} is the degree of vertex i in T^k. At each step of the (revised) simplex method, the variable that enters the basis is determined by minimizing, over all k, the quantity c_k-z_k, where z_k is the inner product of a vector of 'shadow prices' with column k. Let the vector of shadow prices be $(\theta, \pi_2, \pi_3, \cdots, \pi_{n-1})$. Thus we seek a 1-tree T^k for which $c_k-\theta+\sum_{j=2}^{n-1}\pi_j v_{jk}$ is a minimum (if the minimum is nonnegative, then the present basic solution is optimum). We note that if each edge $\{i, j\}$ is assigned a weight $c_{ij}+\pi_i+\pi_j$, with $\pi_1=\pi_n=0$, then the weight of T^k is

$$c_k+\sum_{j=2}^{n-1}\pi_j d_{jk}=c_k+\sum_{j=2}^{n-1}\pi_j(v_{jk}+2)=c_k+\sum_{j=2}^{n-1}\pi_j v_{jk}+2\sum_{j=2}^{n-1}\pi_j.$$

Hence, the column to enter the basis can be determined by a minimum 1-tree calculation with respect to the weights $c_{ij}+\pi_i+\pi_j$. Thus, despite the large number of variables in program (2), the application of the simplex method is quite convenient. Since there are only $n-1$ constraints, the basis is only $(n-1)\times(n-1)$. The minimum 1-tree calculation generates a column to enter the basis in only $O(n^2)$ steps, without the necessity of enumerating or storing the nonbasic columns. An initial basic feasible solution may easily be obtained by constructing $n-1$ 'wheel-like' 1-trees, and assigning the value $1/(n-1)$ to the associated variables y_k. Figure 5 illustrates the construction in the case $n=6$.

This column-generation procedure was programmed for the IBM/360 using standard means of avoiding error propagation: i.e., double-precision floating-point arithmetic, periodic basis reinversion, and checks for zero. The program was able to solve most problems with $n=12$ and some problems with $13\leqq n\leqq 20$. On larger problems, convergence was always too slow to permit optimal solutions to be reached. This slow convergence is consistent with the behavior of other column-generation techniques, which in practice yield good approximate solutions, rather than strictly optimum ones.[9]

Despite the slow convergence, some potentially useful information can be gleaned. It was seen in section 2 that tours correspond to feasible solutions of program (4) in which every x_{ij} is zero or one. Also, every feasible solution (y_k) of program (2) corresponds to a feasible solution (x_{ij}) of program (4) having the same value of the objective function as follows: $x_{ij}=\sum_k y_k e_{ij}^k$. It was observed that, as the simplex method produced a sequence $(y_k^1), (y_k^2), \cdots, (y_k^r)\cdots$ slowly converging to an optimum solution of program (2), most components of the corresponding sequence $(x_{ij}^1), (x_{ij}^2), \cdots, (x_{ij}^r)\cdots$ approached zero or one. A heuristic procedure was tried which successively reduces the problem by eliminating a vertex i

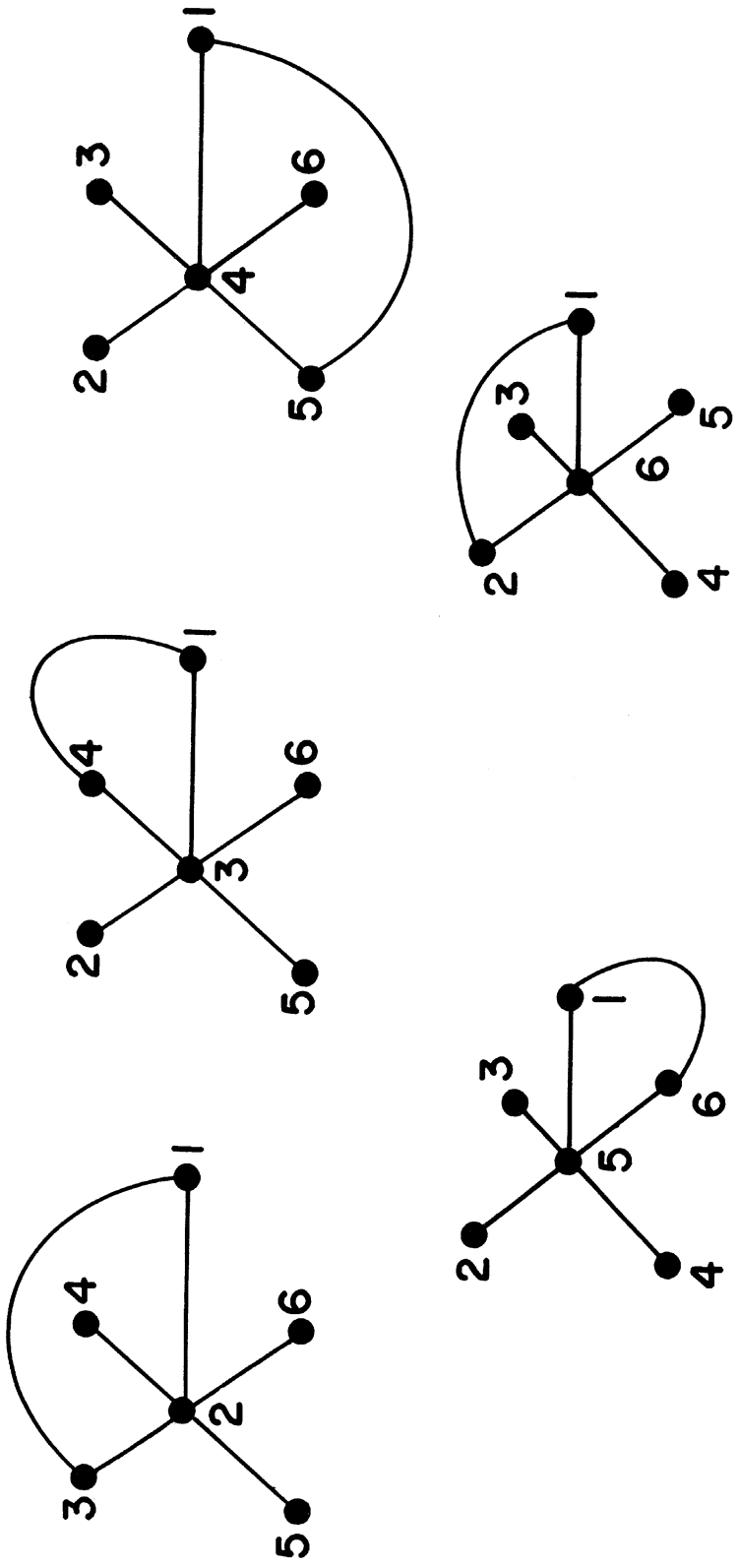


Figure 5.

whenever the proper choice of its two neighbors seems evident from the sequence $\{(x_{ij}^r)\}$. The procedure showed promise on examples up to $n=48$, but was not explored systematically.

4. AN ASCENT METHOD

BECAUSE OF THE slow convergence of the column-generation technique, it is desirable to consider other methods for minimizing $f(\pi)$. It was observed in section 2 that minimizing $f(\pi)$ is equivalent to finding $\max_\pi w(\pi)$, where

$$w(\pi)=\min_k (c_k+\sum_{i=1}^{i=n} \pi_i v_{ik}).$$

In this section, we describe an ascent method for seeking this maximum. Since $w(\pi)$ is the minimum of a finite set of linear (hence concave) functions, it is concave, so that any local maximum is a global maximum.

We derive necessary and sufficient conditions for π to be a maximum point. Let $K(\pi)$ be the set of indices of 1-trees that are of minimum weight with respect to the weights $\bar{c}_{ij}=c_{ij}+\pi_i+\pi_j$. Thus

$$K(\pi)=\{k|w(\pi)=c_k+\sum_{i=1}^{i=n} \pi_i v_{ik}\}.$$

THEOREM 3. *The following are equivalent*:

(1) *π is not a maximum point of the function w.*

(2) *There exists a vector $d=(d_1, d_2, \cdots, d_n)$ such that, for all $k\epsilon K(\pi)$, $\sum_{i=1}^{i=n} d_i v_{ik}>0$.*

Proof. (2)$\Rightarrow$(1). Let d be the vector whose existence is asserted in (2). For any sufficiently small positive ϵ,

$$\begin{aligned} w(\pi+\epsilon d) &= \min_{k\epsilon K(\pi)} (c_k+\sum_{i=1}^{i=n} \pi_i v_{ik}+\epsilon\sum_{i=1}^{i=n} d_i v_{ik}) \\ &= w(\pi)+\epsilon \min_{k\epsilon K(\pi)} \sum_{i=1}^{i=n} d_i v_{ik} > w(\pi). \end{aligned} \tag{6}$$

(1)$\Rightarrow$(2). We assume the negation of (2), and prove that (1) is false. By the Minkowski-Farkas lemma (reference 16, p. 376), the falsity of (2) implies the existence of nonnegative numbers α_k such that $\sum_{k\epsilon K(\pi)} \alpha_k=1$ and $\sum_{k\epsilon K(\pi)} v_{ik}\alpha_k=0$, $i=1, 2, \cdots, n$. Setting $y_k=\alpha_k$, $k\epsilon K(\pi)$, and $y_k=0$, $k\notin K(\pi)$, gives a feasible solution of program (2). The cost of this solution is

$$\sum_{k\epsilon K(\pi)} \alpha_k c_k=\sum_{k\epsilon K(\pi)} \alpha_k(c_k+\sum_{i=1}^{i=n} \pi_i v_{ik})=w(\pi)\sum_{k\epsilon K(\pi)} \alpha_k=w(\pi).$$

But, in view of the duality of programs (1) and (2), we have the following statement:

For any feasible solution $\{y_k\}$ to program (2), $\max_\sigma w(\sigma)\leqq\sum_{K=1}^{k=q} c_k y_k$. Hence $w(\pi)\geqq\max_\sigma w(\sigma)$, and π is a maximum point of $w(\pi)$. This completes the proof.

A vector d satisfying condition (2) of Theorem 3 will be called a *direction of ascent* at π. Let d be such a direction of ascent, and let

$$\Delta(\pi, d) = \min_{k \in K(\pi)} \sum_{i=1}^{i=n} d_i v_{ik}.$$

Then, let

$$K(\pi, d) = \{k | k \in K(\pi) \quad \text{and} \quad \sum_{i=1}^{i=n} d_i v_{ik} = \Delta(\pi, d)\}.$$

Then, it follows from (6) that, for all sufficiently small positive ϵ,

$$K(\pi + \epsilon d) = K(\pi, d) \quad \text{and} \quad w(\pi + \epsilon d) = w(\pi) + \epsilon \Delta(\pi, d).$$

Thus, at a sufficiently small distance from π in the direction d, the function w has increased, and the minimum 1-trees are the ones that are minimum at π and, among 1-trees minimum at π, have the least rate of increase of weight in the direction d.

We define a quantity $\epsilon(\pi, d)$ that determines the maximum distance in the direction d for which this behavior is exhibited: let $\epsilon(\pi, d) = \max\{\epsilon |$ for $\epsilon' < \epsilon$, $K(\pi + \epsilon' d) = K(\pi, d)\}$.

A strategy for finding the maximum of $w(\pi)$ now suggests itself: starting with an arbitrary vector π^0, compute a sequence of vectors $\{\pi^r\}$ as follows: given π^r, find a direction of ascent d^r at π^r and set $\pi^{r+1} = \pi^r + \epsilon(\pi^r, d^r) d^r$.

Determining a direction of ascent at π. Consider the following iteration scheme:

1. Set d equal to the zero n-vector.
2. Find a 1-tree T^k such that $k \in K(\pi, d)$. [A method of executing Step 2 follows from the results of Section 6 (the greedy algorithm).]
3. If $\sum_{i=1}^{i=n} d_i v_{ik} > 0$, STOP.
4. $d_i \leftarrow d_i + v_{ik}, \qquad i = 2, 3, \cdots, n.$
5. GO TO 2.

If a direction of ascent at π exists, then the iteration terminates and yields such a direction d; this follows from a well-known result in pattern recognition (*see*, for example, reference 10). This iteration scheme seems to work well in practice. When no direction of ascent exists [this occurs only at a maximum point of $w(\pi)$], the iteration is nonterminating. Thus, when failure to terminate is suspected, it is necessary to check whether no direction of ascent exists; by the Minkowski-Farkas lemma this is equivalent to the existence of nonnegative coefficients α_k such that

$$\sum_{k \in K(\pi)} \alpha_k v_{ik} = 0, \qquad i = 1, 2, \cdots, n.$$

This can be checked by linear programming.

Determining $\epsilon(\pi, d)$. Let T be a 1-tree. Let e be an edge in T and e', an edge not in T. Then e' is a *substitute* for e in T if $(T - \{e\}) \cup \{e'\}$ is a 1-tree. The edges $e = \{r, s\}$ and $e' = \{i, j\}$ are said to *cross over* at ϵ if the ordered pairs $(\bar{c}_{ij}, d_i + d_j)$ and $(\bar{c}_{rs}, d_r + d_s)$ are unequal but

$$\bar{c}_{ij} + \epsilon(d_i + d_j) = \bar{c}_{rs} + \epsilon(d_r + d_s).$$

THEOREM 4. *Let k be any element of $K(\pi, d)$, where d is a direction of ascent at π. Then $\epsilon(\pi, d) = \min\{\epsilon |$ for some pair (e, e'), e' is a substitute for e in T^k and e and e' cross over at $\epsilon\}$.*

The proof of Theorem 4 depends on concepts developed in Section 6, and is given in that section. To derive an algorithm for computing $\epsilon(\pi, d)$ from this theorem, we need a means of determining when e' is a substitute for e in a 1-tree T. This is the case if and only if either e and e' are both incident with vertex 1 or e is in a cycle of $T \cup \{e'\}$ that does not pass through vertex 1.

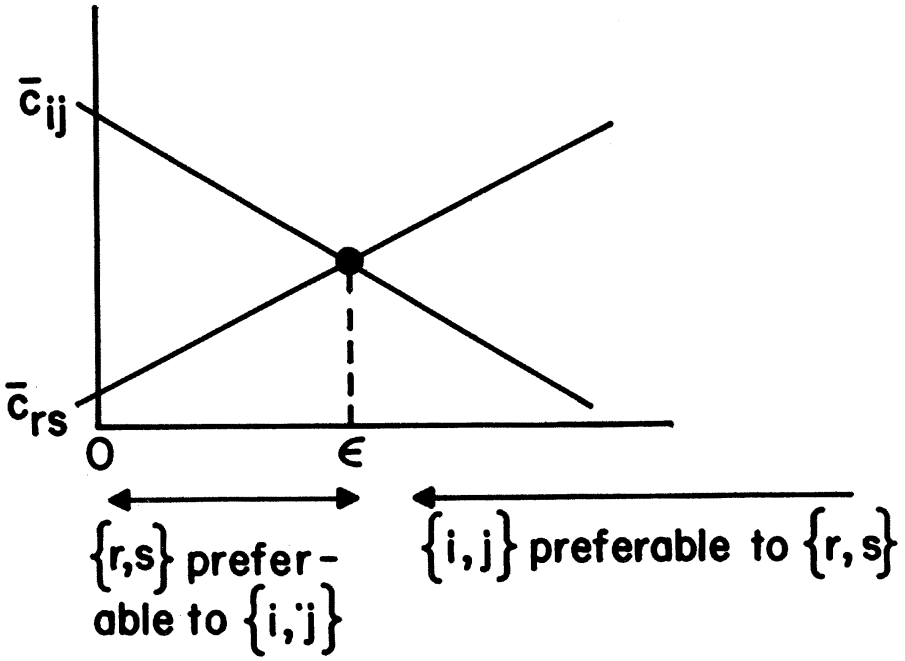


Fig. 6. A crossover.

Performance of the ascent method. A version of the ascent method was programmed and tested on some small problems ($n \leqq 25$). The time per iteration was small, but the number of iterations required grew rapidly with n. Moreover, it has not been determined whether the ascent method always converges to a maximum point of $w(\pi)$.

5. A BRANCH-AND-BOUND METHOD

IN VIEW OF the difficulties encountered with the column-generation and ascent methods, we consider still another approach, in which the ascent method is embedded within a branch-and-bound procedure. The procedure is designed to produce an optimal tour in all cases, even when $\min_\pi f(\pi) > 0$.

Central to this approach is the concept of an out-of-kilter vertex. Ver-

tex i is said to be *out-of-kilter high* at the point π, if, for all $k \epsilon K(\pi)$, $v_{ik} \geqq 1$; similarly, vertex i is *out-of-kilter low* at the point π if, for all $k \epsilon K(\pi)$, $v_{ik} = -1$. Thus, a vertex is out of kilter if its degree in minimum 1-trees is either consistently too high or consistently too low.

Let u_i denote the n-dimensional unit vector with 1 in the ith coordinate. Clearly u_i is a direction of ascent precisely when vertex i is out-of-kilter high, and $-u_i$ is a direction of ascent precisely when vertex i is out-of-kilter low. To determine whether vertex i is out-of-kilter high, one simply finds an arbitrary member k of $K(\pi, u_i)$; i is out-of-kilter high if and only if $v_{ik} \geqq 1$. Similarly, one can test whether i is out-of-kilter low.

The determination of $\epsilon(\pi, d)$ when d is u_i or $-u_i$ is particularly simple. We cite the following corollaries of Theorem 4.

COROLLARY 3. *Assume vertex i is out-of-kilter low and let k be an element of $K(\pi, -u_i)$. Then $\epsilon(\pi, -u_i) = \min(\bar{c}_{ij} - \bar{c}_{rs})$ such that $\{i, j\}$ is a substitute for $\{r, s\}$ in T^k and $i \notin \{r, s\}$.*

COROLLARY 4. *Assume vertex r is out-of-kilter high. Then $\epsilon(\pi, u_r) = \min(\bar{c}_{ij} - \bar{c}_{rs})$ such that $\{i, j\}$ is a substitute for $\{r, s\}$ in T^k, and $r \notin \{i, j\}$.*

Note that, if the $\bar{c}_{ij}$ are integers and d is $\pm u_i$, then $\epsilon(\pi, d)$ is an integer, the new edge weights $[\bar{c}_{ij} + \epsilon(\pi, d)(d_i + d_j)]$ are integers, and $w(\pi + \epsilon d)$ exceeds $w(\pi)$ by an integral amount. Thus, if the ascent method is applied to a problem with integer weights and every direction of ascent is derived from an out-of-kilter vertex, then all weights encountered will be integers, and the value of the function $w(\pi)$ will increase by at least 1 at each iteration. However, in a typical computation a point π^r is reached such that no vertex is out of kilter, but π^r is not a maximum point of $w(\pi)$; i.e., there is a direction of ascent, but not along a unit vector or its negative. Thus, a more complicated vector is required, and fractional weights are introduced. It is at this point that a slowing down of convergence seems to occur in the ascent method. We propose as an alternative that a branch-and-bound technique be applied whenever a point is reached where there is no out-of-kilter vertex.

The following definitions lay the groundwork for the branch-and-bound method. Let X and Y be disjoint sets of edges. Let $T(X, Y)$ denote the set of indices of 1-trees which include all the edges in X and none of the edges in Y. Define $w_{X,Y}(\pi) = \min_{k \epsilon T(X,Y)} (c_k + \sum_{i=1}^{i=n} \pi_i v_{ik})$. Let

$$K_{X,Y}(\pi) = \{k | c_k + \sum \pi_i v_{ik} = w_{X,Y}(\pi)\}.$$

We relativize the concept of an out-of-kilter vertex in the obvious way: vertex i is *out-of-kilter low* at π (relative to X and Y) if, for every $k \epsilon K_{X,Y}(\pi)$, $v_{ik} = -1$; *out-of-kilter high* is similarly defined.

We construct an implicit enumeration technique that finds a minimum tour. At a general step, the state of the computation is given by a list,

each of whose entries specifies a pair (X, Y) of disjoint sets of edges, an integer n-vector π and the number $w_{X,Y}(\pi)$, which is called the *bound* of the entry. Initially the list contains the single entry $[\phi, \phi, 0, w(0)]$. At a general step, an entry $[X, Y, \pi, w_{X,Y}(\pi)]$ of least bound is considered. It is determined whether any vertex is out of kilter at π, relative to X, Y. If so, then a step of the ascent method is carried out, yielding a new entry $[X, Y, \pi', w_{X,Y}(\pi')]$ such that $w_{X,Y}(\pi') \geqq 1 + w_{X,Y}(\pi)$, and the entry $[X, Y, \pi, w_{X,Y}(\pi)]$ is deleted. If no out-of-kilter vertex exists, it is determined whether a direction of ascent exists. If so, then, instead of performing an iteration of the ascent method that would, in general, introduce fractions and improve the objective function by only a small amount, the process 'branches' in a manner described below. If no direction of ascent exists, then a search is conducted to determine whether $K_{X,Y}(\pi)$ includes the index of a tour. If so, the process stops; otherwise, branching is performed.

The process of branching from an entry $[X, Y, \pi, w_{X,Y}(\pi)]$ is as follows. An edge $e \notin X \cup Y$ is chosen, and the entry is replaced by two entries:

$$[X \cup \{e\}, Y^*, \pi, w_{X \cup \{e\}, Y^*}(\pi)] \quad \text{and} \quad [X^*, Y \cup \{e\}, \pi, w_{X^*, Y \cup \{e\}}(\pi)],$$

where Y^* is Y augmented by the edges that cannot occur in any tour which includes $X \cup \{e\}$, and X^* is X augmented by the edges that must occur in any tour that excludes $Y \cup \{e\}$. (In practice only the 'obvious' augmentations will generally be used.)

Let $C_{X,Y}$ denote the minimum weight of a tour that includes the edges in X and excludes the edges in Y. (Minimizing over the empty set always yields the value $+\infty$.) Then, clearly, $\max_\pi w_{X,Y}(\pi) \leqq C_{X,Y}$. Also, because of the way X^* and Y^* are formed, either $\max_\pi w_{X \cup \{e\}, Y^*} \leqq C_{X,Y}$ or $\max_\pi w_{X^*, Y \cup \{e\}} \leqq C_{X,Y}$. Applying this observation inductively, we find that, at any stage of the computation, some entry in the list has a bound less than or equal to W, the weight of a minimum tour. Also, we note the following monotonicity property: $w_{X,Y}(\pi) \leqq w_{X \cup \{e\}, Y^*}(\pi)$ and $w_{X,Y}(\pi) \leqq w_{X^*, Y \cup \{e\}}(\pi)$.

The entire computation can be viewed as a rooted tree whose nodes are all the entries generated. An entry has one direct descendant if it leads to a step of the ascent method because of an out-of-kilter vertex; two direct descendants if it leads to branching; and no descendants if it is never chosen as an entry of least bound. Every entry so chosen has a bound less than or equal to W. Along any path through the tree, each step either enlarges $X \cup Y$ or increases the bound by an integral amount. Hence, every path is finite, and the tree is finite. Thus, the process terminates, and it can only do so by finding a tour of weight less than or equal to W, i.e., a minmum tour.

The branch-and-bound method has been tested only by hand computation. An example follows.

Example 1. The weights c_{ij} are:

1	97	60	73	17	52
2	41	52	90	30	
3	21	35	41		
4	95	46			
5	81				
6					

Initially the list contains the single entry $\phi, \phi, 0, w(0)$.

$\phi, \phi, 0, 196$. There are two minimum 1-trees at $\pi = 0$, and they are displayed in Fig. 7. The 1-trees differ according to which dotted edge is chosen; such a convention will be used in other cases where edge weights are tied and choices are

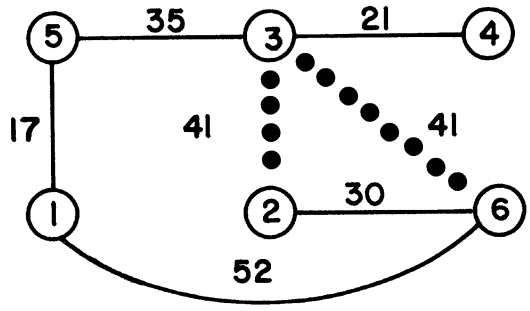


Figure 7.

possible; $w(0) = 196$. Vertex 4 is out-of-kilter low, so $d = -u_4$ is a direction of ascent and $\epsilon(\pi, d) = c_{46} - c_{23} = 5$.

$\phi, \phi, (0, 0, 0, -5, 0, 0), 201$. Here, the $\bar{c}_{ij}$ are:

1	97	60	68	17	52
2	41	47	90	30	
3	16	35	41		
4	90	41			
5	81				
6					

See Fig. 8. There is no out-of-kilter vertex but (0, −1, 1, 0, 0, 0) is a direction of ascent. We branch on edge {2, 3}, obtaining the list: $\{\{2, 3\}\}$, ϕ, (0, 0, 0, −5, 0, 0), 201; ϕ, $\{\{2, 3\}\}$, (0, 0, 0, −5, 0, 0), 201. We consider the first of these.

$\{\{2, 3\}\}, \phi, (0, 0, 0, -5, 0, 0), 201$. *See* Fig. 9. Vertex 3 is out-of-kilter

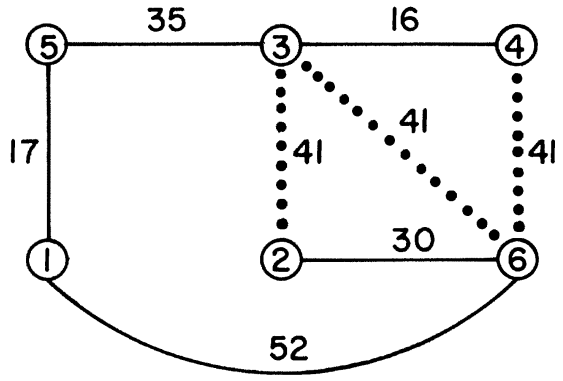


Figure 8.

high; $d=u_3$, $\epsilon(\pi, d) = \bar{c}_{46}-\bar{c}_{34}=25$, and the entry is replaced by $\{\{2,3\}\}$, ϕ, (0, 0, 25, -5, 0, 0), 226.

ϕ, $\{\{2,3\}\}$, $(0,0,0,-5,0,0)$, *201*. *See* Fig. 10. Vertex 2 is out-of-kilter low; $d=-u_2$, $\epsilon(\pi, d)=\bar{c}_{24}-\bar{c}_{46}=6$.

ϕ, $\{\{2,3\}\}$, $(0,-6,0,-5,0,0)$, *207*. The weights $\bar{c}_{ij}$ are:

1	91	60	68	17	52
2	35	41	84	24	
3	16	35	41		
4	90	41			
5	81				
6					

See Fig. 11. There is no out-of-kilter vertex or direction of ascent. The minimum 1-tree in Fig. 12 is a tour, and hence, an optimum tour for the (original) traveling-salesman problem. The tree diagram in Fig. 13 summarizes the calculation.

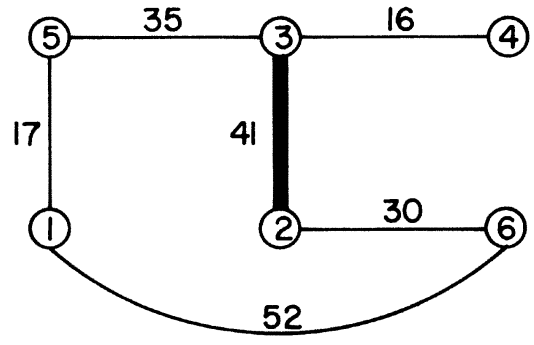


Figure 9.

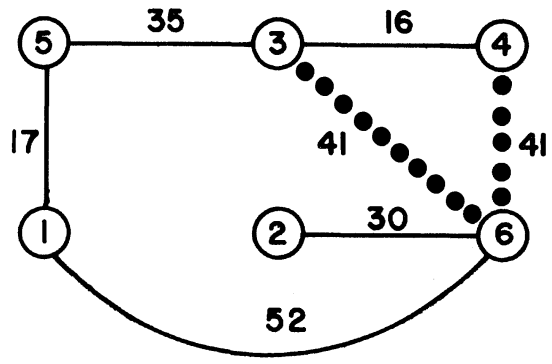


Figure 10.

6. MINIMAL 1-TREES AND MATROIDS

A *matroid*[17] is a structure specified by giving a finite set E together with a specification of each subset of E as *independent* or *dependent* (i.e., not independent), such that (i) any subset of an independent set is independent, (ii) for any set $E' \subseteq E$ the maximal independent subsets of E' all have the same number of elements. This number is called the *rank* of E', and is denoted $r(E')$. A maximal independent subset of E is called a *base*. A minimal dependent subset of E is called a *circuit*.

In this section we outline the theory of minimal 1-trees within the context of matroids, omitting proofs. We also use matroid concepts to prove Theorem 1 (cf. section 2) and Theorem 4 (cf. section 4).

Of particular interest to us is a matroid T whose elements are the edges of K_n. A set of edges is independent in T if it is a subset of a 1-tree; i.e., it contains at most two edges incident with vertex 1 and no cycle is formed by the edges not incident with vertex 1. The bases of T are 1-trees, regarded as sets of edges. A set of edges is a circuit of T if either it consists of exactly three edges, all incident with vertex 1, or it consists of the edges of a cycle that does not pass through vertex 1.

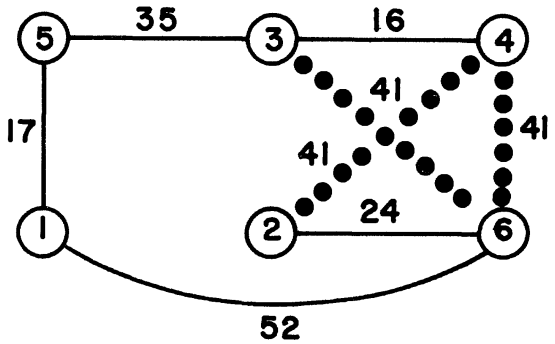


Figure 11.

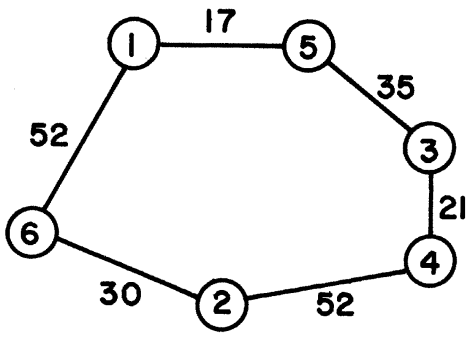


Figure 12.

We discuss in matroid language a slight generalization of the problem of computing a minimum-weight 1-tree. Let E be the set of elements of a matroid, and let $\preccurlyeq$ be an ordering on E which is reflexive, transitive and without incomparable elements ($e \preccurlyeq e'$ or $e' \preccurlyeq e$). The particular orderings of T that interest us are the following:

(i) Numerical ordering: given numerical weights $c_{ij}+\pi_i+\pi_j$ on the edges, $\{i, j\} \preccurlyeq \{r, s\}$ if $c_{ij}+\pi_i+\pi_j \leqq c_{rs}+\pi_r+\pi_s$.

(ii) Lexicographic ordering: given, in addition to numerical weights, a n-vector d, $\{i, j\} \preccurlyeq \{r, s\}$ if $c_{ij}+\pi_i+\pi_j < c_{rs}+\pi_r+\pi_s$ or $c_{ij}+\pi_i+\pi_j = c_{rs}+\pi_r+\pi_s$ and $d_i+d_j \leqq d_r+d_s$.

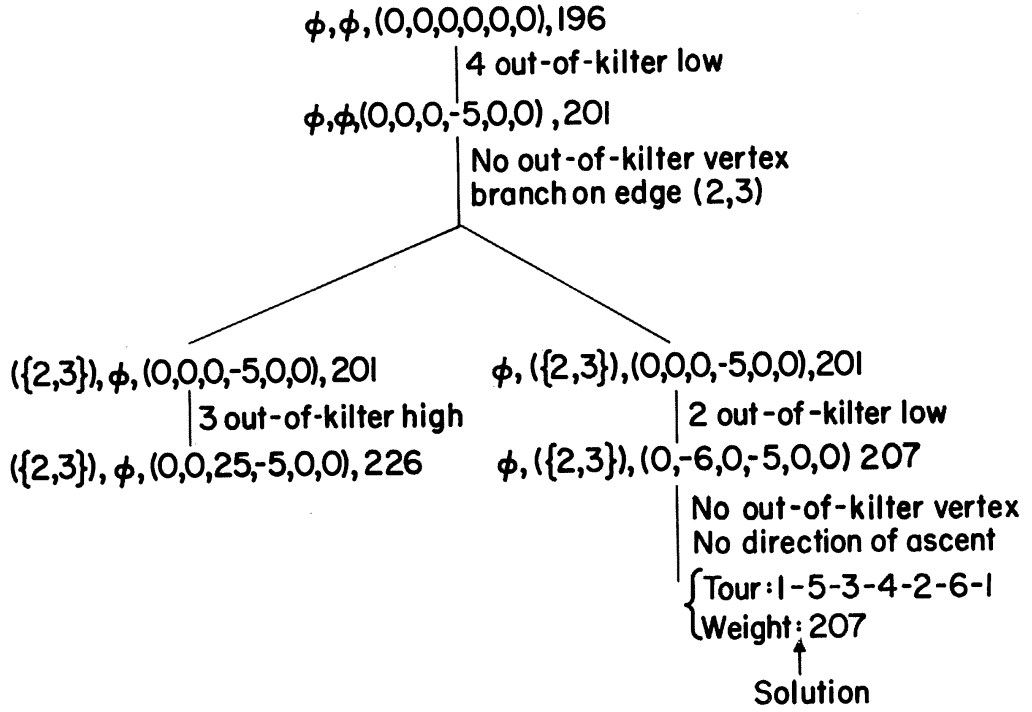


Figure 13.

A base B is called *minimal* (with respect to $\leqslant$), if for any base B', there is a one-one correspondence between the elements of B and those of B' such that, if $e \epsilon B$ corresponds to $e' \epsilon B'$, then $e \leqslant e'$. The minimal 1-trees with respect to numerical ordering are precisely those with indices in $K(\pi)$, and the minimal 1-trees with respect to lexicographic ordering are precisely those with indices in $K(\pi, d)$. For any matroid, and any ordering $\leqslant$, a minimal base exists. The minimal base is unique if $\leqslant$ is a proper ordering (i.e., antisymmetric).

Let $L(e) = \{e' | e' \leqslant e\}$. Then the base B is minimal if and only if, for every e, $B \cap L(e)$ is a maximal independent subset of $L(e)$. This suggests an algorithm for constructing a minimal base. Let $L(e_1) \subset L(e_2) \subset \cdots \subset E$ be the distinct sets $L(e)$ as e ranges over E. Choose a maximal independent subset of $L(e_1)$; augment it to a maximal independent subset of $L(e_2)$; and so on until a base is obtained. This algorithm (usually discussed on the assumption that $\leqslant$ is a proper ordering relation) has been called the *greedy algorithm.*[6,8] It is extremely efficient once E is 'sorted' in accordance with the relation $\leqslant$. In the case of 1-trees with numerical or lexicographic ordering, a method due to DIJKSTRA[4] is preferable unless the sorting has been done in advance.

We now cite a theorem about matroids that, when applied to T, yields Theorem 1 as a special case. For any set $E' \subseteq E$, define span $E' = E' \cup \{e | E' \cup \{e\}$ includes a circuit containing $e\}$; E' is called a *span* if $E' = \text{span}(E')$. If A is independent then $r(\text{span}(A)) = r(A)$. Associate with each element e of E a variable x_e. For each base B, define the incidence vector of B as the following assignment of values to the x_e:

$$x_e = \begin{cases} 1, & e \epsilon B, \\ 0, & e \notin B. \end{cases}$$

THEOREM. *The incidence vectors of the bases are precisely the extreme points of the following system of linear inequalities*: $x_e \geqq 0$, $\sum_{e \epsilon E} x_e = r(E)$, *and* $\sum_{e \epsilon E'} x_e \leqq r(E')$, *for every span* E'.

To the authors' knowledge, this theorem, which is due to JACK EDMONDS, has not appeared in print. Proofs can be found in unpublished papers.[6,13]

In the matroid T, the bases are the 1-trees and the spans are: (i) the set of all edges incident with vertex 1 (this set has rank 2), and (ii) for each set $S \subseteq \{2, \cdots, n\}$, the set of all edges that run between vertices in S (such a set has rank $|S| - 1$). Thus, Theorem 1 is apparent.

Proof. The theorem is equivalent to showing that, for any assignment of rational numbers c_e to the elements e, $\sum c_e x_e$ is maximized, subject to the constraints, by the incidence vector of some base. This is a linear pro-

gram whose dual is the following program, which has a variable y_S for each span $S \subseteq E$:

$$\min \sum y_S r(S), \quad \text{subject to:} \quad \sum_{\{S \mid e \in S\}} y_S \geqq c_e, \quad e \in E; \quad y_S \geqq 0, \quad S \neq E.$$

Suppose the greedy algorithm is applied, choosing elements with largest value of c_e first, since we are maximizing. Let the base B chosen in this way be $\{e_1, e_2, \cdots, e_r\}$ with $c_{e_1} \geqq c_{e_2} \geqq \cdots \geqq c_{e_r}$. The incidence vector of B is a feasible solution of the primal program. Define $S_i = \text{span}(\{e_1, e_2, \cdots, e_i\})$; then $S_r = E$. Defining $y_E = c_r$, $y_{S_i} = c_i - c_{i+1}$, $i = 1, 2, \cdots, r-1$, and $y_S = 0$ otherwise, we obtain a feasible solution of the dual whose cost is easily verified to be equal to $\sum_{e \in B} c_e x_e$. Hence, (by the duality theorem) the incidence vector of B is an optimum solution of the primal program, and the proof is complete.

We now prove Theorem 4.

THEOREM 4. *Let k be any element of $K(\pi, d)$, where d is a direction of ascent at π. Then*

$$\epsilon(\pi, d) = \min\{\epsilon \mid \text{for some pair } (e, e'),$$

$$e' \text{ is a substitute for } e \text{ in } T^k \text{ and } e \text{ and } e' \text{ cross over at } \epsilon\}. \quad (7)$$

Proof. Suppose e' is a substitute for e in T^k and e and e' cross over at ϵ. Then, for any $\epsilon' > \epsilon$, $T^k \notin K(\pi + \epsilon' d)$, since $(T^k - \{e\}) \cup \{e'\}$ is of lower weight than T^k with respect to the weights $\bar{c}_{ij} + \epsilon'(d_i + d_j)$. Thus $\epsilon(\pi, d)$ is less than or equal to the right-hand side of (7). We show the reverse inequality by exhibiting a crossover of the required type at $\epsilon(\pi, d)$. Let

$$\epsilon^* = \max\{0 \cup \{\epsilon \mid 0 < \epsilon < \epsilon(\pi, d) \text{ and some pair of edges cross over at } \epsilon\}\}.$$

Since $\epsilon^* < \epsilon(\pi, d)$, $K(\pi, d) = K(\pi + \epsilon^* d, d)$.

We consider the following two ordering relations on the edges: $\leqslant^*$, lexicographic ordering according to $[\bar{c}_{ij} + \epsilon^*(d_i + d_j), d_i + d_j]$, and $\leqslant$, numerical ordering according to $\bar{c}_{ij} + \epsilon(\pi, d)(d_i + d_j)$. Note that, for any pair (e, e'), (i) $e \leqslant^* e'$ implies $e \leqslant e'$, and (ii) if $e \leqslant e'$ but not $e \leqslant^* e'$, then e and e' cross over at $\epsilon(\pi, d)$. The minimal 1-trees with respect to $\leqslant^*$ are called *-minimal, and are precisely the 1-trees with indices in $K(\pi, d)$. The minimal 1-trees with respect to $\leqslant$ are merely called minimal; they are the 1-trees with indices in $K[\pi + \epsilon(\pi, d)d]$. Every *-minimal 1-tree is also minimal, but not conversely.

We consider an arbitrary *-minimal 1-tree T^k and an arbitrary minimal but not *-minimal 1-tree T. Our object is to show that there exist an edge e' in T, and an edge e in T^k such that e' substitutes for e in T^k and e and e' cross over at $\epsilon(\pi, d)$; i.e., e is in a circuit of $T^k \cup e'$, $e' \leqslant e$ and $e' \not\leqslant^* e$. We use the characterization of minimal (or *-minimal) bases which led to the greedy algorithm. For any e, let $L(e) = \{e' \mid e' \leqslant e\}$ and let $L^*(e) = \{e' \mid e' \leqslant^* e\}$. Let the distinct sets $L^*(e)$ as e ranges over E be

$$L_1^* \subset L_2^* \subset \cdots \subset L_j^* \subset \cdots \subset E, \tag{8*}$$

and let the distinct sets $L(e)$ as e ranges over E be

$$L_1 \subset L_2 \subset \cdots \subset L_m \subset \cdots \subset E. \tag{8}$$

But any set $L(e)$ is also of the form $L^*(f)$; simply choose $f \epsilon L(e)$ such that $e' \leqslant^* f$ for all $e' \epsilon L(e)$. Hence the sequence (8) is a subsequence of (8*) and can be written

$$L_{i_1}^* \subset L_{i_2}^* \subset \cdots \subset L_{i_p}^* \subset \cdots \subset E.$$

Now, since T^k is *-minimal, $|T^k \cap L_j^*| = r(L_j^*)$ for all j, and, since T is minimal but not *-minimal, $|T \cap L_{i_p}^*| = r(L_{i_p}^*)$ for all p, but $|T \cap L_j^*| < r(L_j^*)$ for some j. Let $s = \min\{j \mid |T \cap L_j^*| < r(L_j^*)\}$, and let t be $\min\{p \mid L_s^* \subseteq L_{i_p}^*\}$. Now a simple argument involving ranks yields the conclusion. For brevity, write

$$\begin{aligned} T \cap L_{s-1}^* &= A, & T \cap (L_s^* - L_{s-1}^*) &= B, & T \cap (L_{i_t}^* - L_s^*) &= C, \\ T^k \cap L_{s-1}^* &= A^k, & T^k \cap (L_s^* - L_{s-1}^*) &= B^k, & T^k \cap (L_{i_t}^* - L_s^*) &= C^k. \end{aligned}$$

Then $A \cup B \cup C$ is independent, and $A^k \cup B^k \cup C^k$ is independent. Also, $|A| = |A^k| = r(L_{s-1}^*)$, $|A \cup B \cup C| = |A^k \cup B^k \cup C^k| = r(L_{i_t}^*)$, and $|A^k \cup B^k| = r(L_s^*) > |A \cup B|$. Hence $|C| > |C^k|$ and $|A \cup C| > |A^k \cup C^k|$, so $A \cup C \not\subseteq$ span $(A^k \cup C^k)$. But $A \subseteq$ span $A^k \subseteq$ span $(A^k \cup C^k)$, so

$$C \not\subseteq \text{span } (A^k \cup C^k). \tag{9}$$

On the other hand,

$$C \subseteq A \cup B \cup C \subseteq \text{span } (A^k \cup B^k \cup C^k). \tag{10}$$

By (9) there is an element $e' \epsilon C$ such that $A^k \cup C^k \cup \{e'\}$ does not include a circuit. But, by (10), $A^k \cup B^k \cup C^k \cup \{e'\}$ does include a (unique) circuit. Hence this circuit must contain some element $e \epsilon B^k$. We conclude that $T^k \cup \{e'\}$ includes a circuit containing e (i.e., e' substitutes for e in T^k) where $e' \epsilon L_{i_t}^* - L_s^*$ and $e \epsilon L_s^* - L_{s-1}^*$. Thus e' is not $\leqslant^* e$, but $e' \leqslant e$, so e and e' cross over at $\epsilon(\pi, d)$. This completes the proof.

7. OTHER APPLICATIONS

THE APPROACH developed in this paper for the symmetric traveling-salesman problem can be applied to other problems. Consider first the nonsymmetric traveling-salesman problem, which asks for a directed tour of minimum weight through n cities. Here the cost of travel from city i to city j may differ from the cost in the reverse direction. In this case, a type of directed graph called a 1-arborescence plays the same role as the 1-tree did in the symmetric case. A 1-arborescence is a directed graph on the vertex set $\{1, 2, \cdots, n\}$ such that exactly one edge is directed into each vertex,

there is exactly one cycle, and the cycle passes through vertex 1. A method of computing a minimum-weight 1-arborescence follows from the work of Edmonds.[7] A minimum tour is the same for the weights $c_{ij}+\pi_i$ as for c_{ij}, but the minimum 1-arborescences change. We seek a $\pi=(\pi_1, \pi_2, \cdots, \pi_n)$ that minimizes the difference between the cost of a minimum tour and the cost of a minimum 1-arborescence. The entire development is easily extended to this case.

Another application concerns a modified symmetric traveling-salesman problem in which cities $2, 3, \cdots, n$ are to be visited by p salesmen, all based at city 1. Then we seek to cover the cities at minimum cost by a p-tour, i.e., a set of p subtours that are disjoint except that each passes through city 1. A typical p-tour has the clover-leaf structure indicated in Fig. 14.

We proceed by relating the given problem to an easier one, the minimum p-tree problem. A p-tree consists of a p-component forest on vertices $2, 3, \cdots, n$, together with two edges from vertex 1 to each component of the forest. A minimum p-tour is the same for the weights c_{ij} as for the

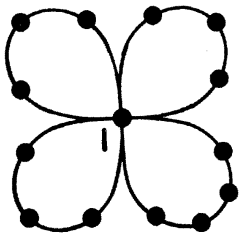


Figure 14.

weights $c_{ij}+\pi_i+\pi_j$, but the minimum p-trees change, and the systematic variation of the π_i proceeds much as in the case of the symmetric traveling-salesman problem.

A single-facility scheduling problem[11] admits of a formulation along similar lines. The problem is as follows: given n jobs with execution times t_i and associated cost functions $c_i(t)$, execute them consecutively, once each, beginning at time 0, so as to minimize $\sum_i c_i(\bar{t}_i)$, where $\bar{t}_i$ is the termination time of job i. Note that, in the case where the t_i are equal to 1, this problem reduces to the assignment problem (*see* below). A simpler related problem is: execute jobs consecutively without gaps throughout the interval $[0, \sum_i t_i]$ so as to minimize the sum of the costs associated with all terminations; the number of executions of each job is left open. This problem can easily be solved by dynamic programming.[5] We note that adding constants π_i to the cost functions $c_i(t)$ does not change the original problem, but does change the related problem. Again, we seek π_i that minimize the difference between the costs for the two problems, and we are led to a gap-

minimization problem that can be treated by the methods of the present paper.

The well known assignment problem can be treated similarly. The problem can be stated as follows: given a $n \times n$ matrix (c_{ij}), choose n entries such that exactly one entry is chosen in each row or column, so as to minimize the sum of the entries chosen. We define an entirely trivial related problem: choose an element from each row of a matrix so as to minimize the sum of the elements chosen. We observe that adding constants π_j to columns does not change the assignment problem, but does change the related problem. We seek π_j that minimize the difference between the costs for the two problems. The well known and efficient Hungarian Method can be interpreted as a descent method for finding this minimum. It can be shown that, in this particular case, the difference can be reduced to zero.

There are results about matroids that give insight into why the gap can be reduced to zero for the assignment problem but not for the traveling-salesman problem. Suppose we have k matroids, each with the set of elements E. Let the rank of E' in matroid i be $r_i(E')$, and assume $r_i(E) = n$ for all i. Associate with each element e a variable x_e, and consider the following system of linear inequalities: $0 \leqq x_e \leqq 1$, $e \epsilon E$; if E' is a span in matroid i, $\sum_{e \epsilon E'} x_e \leqq r_i(E')$; $\sum_{e \epsilon E} x_e = n$. When k is 2, the extreme points of this system are precisely the incidence vectors of sets E' that are bases in the two matroids. The analogous statement for $k \geqq 3$ is false, in general.

Now consider the following three definitions of independence, each determining a matroid over the set $\{1, 2, \cdots, n\} \times \{1, 2, \cdots, n\}$ of all directed edges between vertices in $\{1, 2, \cdots, n\}$:

1. E' is independent if no two edges in E' are directed to the same vertex.

2. E' is independent if no two edges in E' are directed from the same vertex.

3. E' is independent if the elements of E', with directions ignored, form an independent set in T, the matroid of 1-trees.

These three matroids have bases in common. The common bases of the *two* matroids 1 and 2 correspond to the feasible solutions of the assignment problem (this explains why the gap is zero), and the common bases of all *three* matroids are the incidence vectors of directed tours (suggesting that the gap is in general nonzero for the traveling-salesman problem).

Note added in proof: Recently an algorithm based on the concepts in this paper has been tested extensively, with excellent results. It is based upon a highly effective new method for computing $\max_\pi \omega(\pi)$, together

with an improved branch-and-bound strategy. Details will be presented in a sequel to the present paper.

ACKNOWLEDGMENTS

THE AUTHORS wish to thank A. J. HOFFMAN and P. S. WOLFE for their helpful suggestions.

Karp's research was partially supported by the National Science Foundation under a grant to the University of California, and part of his work was done at the IBM Thomas J. Watson Research Center.

REFERENCES

1. BELLMORE, M. AND G. L. NEMHAUSER, "The Traveling Salesman Problem: A Survey," *Opns. Res.* **16,** 538–558 (1968).
2. DANTZIG, G. B., D. R. FULKERSON, AND S. M. JOHNSON, "Solution of a Large Scale Traveling Salesman Problem," *Opns. Res.* **2,** 393–410 (1954).
3. ———, ———, AND ———, "On a Linear Programming, Combinatorial Approach to the Traveling Salesman Problem," *Opns. Res.* **7,** 58–66 (1959).
4. DIJKSTRA, E. W., "A Note on Two Problems in Connexion with Graphs," *Numerische Mathematik* **1,** 269–271 (1959).
5. EASTMAN, W., "A Note on the Multi-Commodity Warehouse Problem," *Management Sci.* **5,** 327–331 (1959).
6. EDMONDS, J., "Matroids and the Greedy Algorithm," to appear.
7. ———, "Optimum Branchings," *J. Res. National Bureau of Standards* **71B,** 233–240 (1967).
8. GALE, D., "Optimal Assignments in an Ordered Set: An Application of Matroid Theory," *J. Combinatorial Theory* **4,** 176–180 (1968).
9. GILMORE, P. C. AND R. E. GOMORY, "A Linear Programming Approach to the Cutting-Stock Problem, Part II," *Opns. Res.* **11,** 863–888 (1963).
10. GREENBERG, H. J. AND A. G. KONHEIM, "Linear and Nonlinear Methods in Pattern Classification," *IBM J. Research and Development* **8,** 299–307 (1964).
11. HELD, M. AND R. M. KARP, "A Dynamic Programming Approach to Sequencing Problems," *SIAM* **10,** 196–210 (1962).
12. KRUSKAL, J. B., "On the Shortest Spanning Subtree of a Graph and the Traveling Salesman Problem," *Proc. Am. Math. Soc.* **1,** 48–50 (1956).
13. LAWLER, E. L., "Optimum Matroid Intersections," to appear.
14. MARTIN, G. T., "Solving the Traveling Salesman Problem by Integer Programming," *C.E.I.R.*, New York (1966).
15. OBRUCA, A. K., "Spanning Tree Manipulation and the Traveling Salesman Problem," *Computer J.* **10,** 374–377 (1968).
16. SIMONNARD, M. (W. JEWELL, translator), *Linear Programming*, Prentice-Hall, New York, 1966.
17. WHITNEY, H., "On the Abstract Properties of Linear Dependence," *Am. J. Math.* **57,** 509–533 (1935).