JOURNAL OF RESEARCH of the National Bureau of Standards—B. Mathematics and Mathematical Physics
Vol. 71B, No. 4, October–December 1967

Optimum Branchings*

Jack Edmonds

Institute for Basic Standards, National Bureau of Standards, Washington, D.C. 20234

To Professor Marcel Riesz on His 80th Birthday

(November 16, 1966)

An arborescence T is a tree whose edges are directed so that each is directed toward a different node. Exactly one node of T, called the root, has no edge of T directed toward it. Let G be any directed graph with a real numerical weight on each edge. A good algorithm is described for finding in G (if there is one) a spanning arborescence, with prescribed root, whose edges have maximum (or minimum) total weight.

Key Words: Algorithms, arborescences, branchings, combinatorics, graphs, linear programming, traveling salesman, trees.

Section 1

A (*directed*) *graph* G, for purposes here, is a finite set of *nodes* and a finite set of *edges*, where each edge is said to be *directed toward* one of the nodes, called the *front end* of the edge, and said to be *directed away from* a different one of the nodes, called the *rear end* of the edge. An edge and each of its ends are said to *meet*. A *subgraph* of G is a subcollection of its members which, under the same incidence relations, is a graph. A graph is called *connected* if it is not empty and its members do not partition into two disjoint nonempty subgraphs. A *polygon* is a connected graph Q such that each node of Q meets exactly two edges of Q. An (*elementary uniformly directed*) *circuit* is a polygon which contains one edge directed toward, and one edge directed away from, each of its nodes. A *forest* is a graph which contains no polygon. A *tree* is a connected forest. A *branching* is a forest whose edges are directed so that each is directed toward a different node. An *arborescence* is a connected branching. An (*elementary uniformly directed*) *path* P is an arborescence such that each edge in P is directed away from a different node, and such that there is at least one edge in P.

We shall occasionally use "obvious" facts about graphs without justifying them.

Clearly, a branching (forest) is the union of a unique family of disjoint arborescences (trees).

Exactly one node in an arborescence T, called the *root* of T, has no edge of T directed toward it. A branching (forest) is an arborescence (tree) if and only if it has exactly one less edge than nodes. No branching (forest) has more edges than this.

In a path P there are exactly two nodes, called the *ends* of P, which each meet only one edge in P. The rest of the nodes in P each meet exactly two edges in P. A path P is said *to go from* the node which is only a rear end in P (the root of P) *to* the node which is only a front end in P. For any arborescence T, and any node v in T except the root, there is a unique path in T going from the root to v. Any path in T going to v and any path in T going from v have only v in common, and their union is a path. And so on.

Section 2

Let G be any graph with a real numerical weight c_j corresponding to each edge $e_j \epsilon G$. The problem treated here is to find in G a branching B which has maximum total weight, Σc_j, summed over $e_j \epsilon B$. B is called an optimum branching in G.

First we show that certain variations of the problem reduce immediately to it.

A *spanning* subgraph of G is a subgraph which contains all the nodes of G. A branching in G is a spanning arborescence of G if and only if the number of its edges is one less than the number of nodes in G. No branching in G can have more edges than this.

An optimum branching in G of course contains no edge with negative weight, and indeed may be empty if all $c_j \leq 0$. Even if all $c_j > 0$ and G contains a spanning arborescence, an optimum branching in G need not be an arborescence.

If there is a spanning arborescence T in G, then an optimum one, i.e., one which has maximum total weight, Σc_j, $e_j \epsilon T$, can be found as an optimum branching in G where the edges carry new weights

*Prepared while the author was a visiting professor at the University of Waterloo, Ontario, Canada. Presented under the title Optimum Arborescences at the International Seminar on Graph Theory and Its Applications, Rome, July 1966.

$$c_j' = c_j + h,\ h > \Sigma \mid c_j \mid,\ e_j \epsilon G.$$

A spanning arborescence in G which is optimum relative to weights c_j, $e_j \epsilon G$, is also optimum relative to weights $c_j + k$, $e_j \epsilon G$, for any constant k, since every spanning arborescence has the same number of edges. Constant h is larger than the difference in total weights (relative to weights c_j, $e_j \epsilon G$) of any two branchings in G. It follows that an optimum branching in G, relative to weights $c_j' = c_j + h$, will be a branching with a maximum number of edges. In particular, it will be a spanning arborescence if and only if G contains a spanning arborescence.

A spanning arborescence T in G which has minimum total weight, Σc_j, $e_j \epsilon T$, is the same as one which has maximum total weight $\Sigma c_j'$, $e_j \epsilon T$, relative to weights $c_j' = -c_j$.

It will be evident that the efficiency of the method for treating optimum branchings is not seriously effected by a large change h (say of the form 10^n) in all the weights. In fact the method is easily modified to treat optimum spanning arborescences directly.

If there is a spanning arborescence in G which is rooted at a prescribed node, say r, then an optimum one can be found by finding an optimum spanning arborescence in the graph G' obtained from G by adjoining a new edge e_0 (carrying arbitrary weight c_0) which is directed toward r and directed from a new node having no other incident edges. Clearly, T is a spanning arborescence in G which is rooted at r if and only if T together with e_0 is a spanning arborescence of G'.

If the edges in graph G represent the links for possible direct communication from one node to another, if each c_j is the cost of direct communication from the rear end of e_j to the front end of e_j, and if cost is additive, then a minimum-total-weight spanning arborescence rooted at prescribed node r represents the least costly way to have a message communicated from r to all other nodes of G.

Another application is where it is desired to arrange an institution into an optimum heirarchy (branchocracy).

Section 3

Our main result is

THEOREM 1. *There exists a good algorithm for finding, in any graph* G *with a numerical weight corresponding to each edge, an optimum branching.*

We say an algorithm is *good* if there is a polynomial function $f(n)$ which, for every positive-integer valued n, is an upper bound on the "amount of work" the algorithm does for any input of "size" n. The concept is easy to formalize—relative, say, to a Turing machine, or relative to any typical digital computer with an unlimited supply of tape.

For optimum branching, the largest number of significant digits in an edge weight, as well as the number of edges of G, must be figured somehow into the measure n of input "size." One might for example take n to be the maximum of these two numbers or to be the vector consisting of both numbers.

The proof of Theorem 1 is constructive. The theorem is proved by displaying one particular algorithm for optimum branching which is obviously good.

If we remove from the optimum-spanning-arborescence problem the condition that each member of the set T of edges being optimized must have a different front end, then we get the optimum-spanning-tree problem. That is to find, if there is one, in any graph G with a numerical weight on each edge, a spanning tree which has maximum (or minimum) total weight.

Especially simple algorithms are well-known for this problem [cf. 5 and 6].[1] One is, starting with an empty bucket, build up a set of elements having "admissible structure" by putting elements into the bucket one after another as long as possible, so that each addition is a maximum weight element among those not in the bucket which, together with the ones already in the bucket, would preserve admissible structure. For the optimum-spanning-tree problem, the elements are the edges of G and "admissible" means "forest." The algorithm is certainly good. It is also valid for that problem.

Where "admissible" means "branching," the above algorithm is not generally valid for finding an optimum spanning arborescence. Paper [3] abstractly characterizes those structures for which this "greedy algorithm" is valid for any numerical weighting.

If we add to the conditions of the optimum-spanning-arborescence problem the condition that each member of the set of edges being optimized is to have a different rear end, then we have the problem of finding, if there is one, an optimum spanning (uniformly directed) path in any graph G with a numerical weight on each edge. This is a version of the well-known traveling saleman problem [cf. 4]. I conjecture that there is no good algorithm for the traveling saleman problem. My reasons are the same as for any mathematical conjecture: (1) It is a legitimate mathematical possibility, and (2) I do not know.

A good algorithm is known for finding, in any graph with a numerical weight on each edge, a maximum-total-weight subset of edges such that no two of them meet the same node [1, 2]. The treatment here of optimum branchings is similar.

Section 4

Here is the algorithm for finding a maximum-total weight branching in any (directed) graph G with a numerical weight c_j on each edge $e_j \epsilon G$. Recall that a branching is a forest such that each edge is directed toward a different node.

Begin the algorithm by applying instruction (I 1) where G^i is $G^0 = G$ and where D^i and E^i are empty buckets, D^0 and E^0.

(I 1) Choose a node v in G^i and not in D^i. Put v into bucket D^i. If there is in G^i a positively weighted edge

[1] Figures in brackets indicate the literature references at the end of this paper.

directed toward ν, put one of them having maximum weight into bucket E^i.

Repeat (I 1) until
(a) E^i no longer comprises the edges of a branching in G^i, or until (b) every node of G^i is in D^i, and E^i does comprise the edges of a branching. When case (a) occurs, apply (I 2).

For convenience assume that every branching which we consider in graph G^i contains all the nodes of G^i. We say that a set of edges in G^i *forms* the unique subgraph of G^i consisting of those edges and all nodes in G^i.

Each edge e put into E^i according to (I 1) is directed toward a node ν which is the root of a connected component of the branching, say B, formed by the edges in E^i before e is put into E^i. If the rear end ν_6 of e is in a different component of B than ν, then $B \cup e$ is a branching, and so when e is put into E^i, (a) does not hold.

If ν_6 is in the same component of B as ν, then B contains a unique path P going from ν to ν_6. In this case, $Q^i = P \cup e$ is a circuit contained in $B \cup e$, so as soon as e is put into E^i, (a) does hold.

(I 2) Store Q^i and a specification of one of the edges, say e_0^i, of Q^i which has minimum weight in Q^i relative to the edge-weights for G^i. Obtain a new graph G^{i+1} from G^i by "shrinking" to a single new node, ν_1^{i+1}, the circuit Q^i and every edge of G^i which has both ends in Q^i. The edges (denoted as e_j^{i+1}) of G^{i+1} are those edges (denoted as e_j^i) of G^i which have at most one end in Q^i. Every edge of G^i which has one end in Q^i will in G^{i+1} have ν_1^{i+1} at that end. All other edge-ends are the same in G^{i+1} as in G^i. The nodes of Q^i are not in G^{i+1}.

Every edge, say e_3^{i+1}, which as e_3^i in G^i is directed toward a node, say ν_3^i, in Q^i and directed away from a node not in Q^i, gets a possibly different weight for G^{i+1}:

$$c_3^{i+1} = c_3^i + c_0^i - c_4^i \tag{1}$$

where c_3^i is the weight of e_3^i for G^i; where c_0^2 is the minimum weight for G^i of an edge, say e_0^i, in Q^i; and where c_4^i is the weight for G^i of the unique edge, say e_4^i, which is in Q^i and directed toward ν_3^i. All other edges in G^{i+1} keep the same weight as for G^i.

In justifying the algorithm we shall make use of the following relations

$$(2)\quad c_0^i \geqslant 0, \qquad (3)\quad c_4^i \geqslant c_0^i, \quad \text{and} \quad (4)\quad c_4^i \geqslant c_3^i.$$

Put into bucket D^{i+1} the nodes which are in both G^{i+1} and bucket D^i. (Do *not* at this point put ν_1^{i+1} into D^{i+1}.) Put into bucket E^{i+1} the edges which are in both G^{i+1} and bucket E^i, i.e., put into bucket E^{i+1} the final contents of bucket E^i minus the edges of circuit Q^i. It is easy to see that the edges in bucket E^{i+1} form a branching in G^{i+1}. Continue the algorithm by applying (I 1) where i is one greater.

Eventually, after a small number of applications of (I 1) and (I 2), case (b) must occur.

As soon as (b) occurs, for say $i=k$, (I 1) and (I 2) are never applied again. Instead, (I 3) is applied successively for $i+1=k,\ k-1,\ \ldots,\ 1$, until the graph G^i obtained is the original G. At that point, the branching $B^i = B^0$ is a maximum-total-weight branching of G.

The final contents of bucket E^k form a branching in graph G^k which we call B^k.

(I 3) It is not difficult to see that since B^{i+1} is a forest in G^{i+1} and since G^{i+1} is obtained from G^i by shrinking the circuit Q^i in G^i (and all edges of G^i with both ends in Q^i) to the node ν_1^{i+1} of G^{i+1}, the subgraph H^i of G^i, formed by the edges in B^{i+1} and the edges in Q^i contains only one polygon, namely Q^i.

In the case where ν_1^{i+1} is not a root of (a connected component of) branching B^{i+1} in G^{i+1}, there is a unique edge, say e_1^{i+1}, of B^{i+1} which is directed toward ν_1^{i+1}. In G^i, e_1^i is directed toward a node, say ν_2^i, of Q^i. Since Q^i is a circuit, there is a unique edge, say e_2^i, of Q^i which is directed toward ν_2^i. Clearly, e_1^i and e_2^i are the only two edges of H^i which are directed toward the same node. Thus, since Q^i is in the only polygon of H^i, deleting e_2^i from H^i yields a branching in G^i, which is called B^i.

In the case where ν_1^{i+1} is a root of branching B^{i+1} in G^{i+1}, i.e., where no edge of B^{i+1} is directed toward ν_1^{i+1}, no two edges of H^i are directed toward the same node. Therefore, deleting any edge of Q^i from H^i yields a branching in G^i. To obtain the branching B^i in G^i, delete from H^i one of the edges e_0^i of Q^i which has minimum weight c_0^i.

That completes the description of the algorithm. Evidently it is a good algorithm. Evidently its output is a branching B^0 in graph G. In order to prove Theorem 1, what remains to be done is prove that B^0 has maximum total weight.

Section 5

Theorem 1 and the following geometric theorem are proven together.

Let G be any graph. (No edge-weights are specified.) Let there be a real variable x_j for each edge $e_j \in G$. Let P_G be the polyhedron of vectors $x=[x_j]$ which satisfy the system L_G, consisting of inequalities L_1, L_2, and L_3.

(L_1) For every edge $e_j \in G$, $x_j \geqslant 0$.
(L_2) For every node $\nu \in G$, $\Sigma x_j \leqslant 1$, summed over all j's such that e_j is directed toward ν.
(L_3) For every set S of two or more nodes in G,

$$\Sigma x_j \leqslant |S| - 1,$$

summed over all j's such that e_j has both ends in S. ($|S|$ denotes the cardinality of S.)

Any vector $x=[x_j]$ of zeroes and ones is called *the (incidence) vector* of the subset of e_j's such that $x_j = 1$.

THEOREM 2. *The vertices of polyhedron* P_G *are precisely the vectors of the subsets of edges in* G *which comprise branchings.*

A *polyhedron* (*convex polyhedron*) P is the set of all the vectors, i.e., *points*, which satisfy some finite system L of linear inequalities. A *vertex* (extreme point) of P is a point which, for some linear function, is the unique point in P which maximizes that function.

A *basic* point $x=x^0$ of a finite system L of linear inequalities is the unique solution of a system, ${}_j\Sigma a_{ij}x_j=b_i$, $i\epsilon I$, such that ${}_j\Sigma a_{ij}x_j \leqslant b_i$, $i\epsilon I$, is a subsystem of L.

If basic point x^0 of L is in the polyhedron P of L, then it is a vertex of P, because clearly x^0 is then the unique point in P which maximizes ${}_j\Sigma({}_i\Sigma a_{ij})x_j$, $i\epsilon I$.

We shall see without difficulty that any point x^0, which is the vector of a branching say B^0 in G, is a vertex of P_G. Vector x^0 satisfies L_1 since it is all zeroes and ones. Vector x^0 satisfies L_2 for any node $\nu\epsilon G$, since, by the definition of branching, at most one of the x_j's in this inequality has value 1 for x^0.

The branching B^0 is a forest, so any set S of nodes, together with the subset E_0^S of the edges in B^0 which have both ends in S forms a forest. The number of edges in a forest is at most the number of nodes in the forest minus 1; in particular, $|E_0^S| \leqslant |S|-1$. Therefore, vector x^0 satisfies L_3 for any subset S of (two or more) nodes in G, since $|E_S^0|$ of the x_j's in this inequality have the value 1 for x^0. Summarizing the conclusion so far, x^0 is a point in P_G.

Vector x^0 is the unique solution of the linear system: $x_j=0$ for every edge e_j not in B^0, and $\Sigma x_j=1$ (summed over e_j's directed toward ν) for every node ν which has some edge of B^0 directed toward it. This system can be obtained from certain of the relations of L_1 and L_2 by replacing their inequality signs. Therefore x^0 is a basic point of L_G, and hence a vertex of P_G.

Most of this paper is directed toward proving:

LEMMA 1: *Every linear function,* $\Sigma c_j x_j$ (*summed over all edges* $e_j\epsilon G$), *is maximized in* P_G *by the vector of some branching in* G.

From Lemma 1 and from the definition of vertex, it follows immediately that every vertex of P_G is the vector of a branching in G. This will conclude the proof of Theorem 2.

A branching B^0 in graph G has maximum total weight relative to the vector $c=[c_j]$ of edge-weights if and only if the vector $x^0=[x_j^0]$ of B^0 maximizes $(c, x)={}_j\Sigma c_j x_j$ over all vectors of branchings in G. If x^0 maximizes (c, x) over P_G, then it maximizes (c, x) over the vectors of branchings in G, since the latter are in P_G.

Our task, therefore, is to show that the vector of the branching B^0, produced by the algorithm, maximizes (c, x) over P_G. This will prove that the algorithm is valid and will prove Lemma 1.

Section 6

The following computations are well-known in linear programming. Suppose that $x=[x_\eta]$ is any vector which satisfies

$$x_\eta \geqslant 0 \text{ for every } \eta, \text{ and} \tag{5}$$

$${}_\eta\Sigma a_{\xi\eta}x_\eta \leqslant b_\xi \text{ for every } \xi, \tag{6}$$

and that $y=[y_\xi]$ is any vector which satisfies

$$y_\xi \geqslant 0 \text{ for every } \xi, \text{ and} \tag{7}$$

$${}_\xi\Sigma a_{\xi\eta}y_\xi \geqslant c_\eta \text{ for every } \eta. \tag{8}$$

Since (6) and (7) imply

$${}_\xi\Sigma({}_\eta\Sigma a_{\xi\eta}x_\xi)y_\xi \leqslant {}_\xi\Sigma b_\xi y_\xi=(b, y), \tag{9}$$

and since (5) and (8) imply

$${}_\eta\Sigma({}_\xi\Sigma a_{\xi\eta}y_\xi)x_\eta \geqslant {}_\eta\Sigma c_\eta x_\eta=(c, x), \tag{10}$$

we have

$$(c, x) \leqslant (b, y). \tag{11}$$

Since (11) holds for any x and any y, if $(c, x^0)=(b, y^0)$ holds for particular $x=x^0$ and $y=y^0$, then x^0 must maximize (c, x) and y^0 must minimize (b, y).

Suppose for particular $x=x^1$ and $y=y^1$ that

$${}_\eta\Sigma a_{\xi\eta}x_\eta^1=b_\xi \text{ for } \xi \text{ such that } y_\xi^1 \neq 0, \tag{12}$$

and

$${}_\xi\Sigma a_{\xi\eta}y_\xi^1=c_\eta \text{ for } \eta \text{ such that } x_\eta^1 \neq 0. \tag{13}$$

Since (12) implies equality in (9), and (13) implies equality in (10), we have $(c, x^1)=(b, y^1)$. Therefore,

$$x^1 \text{ maximizes } (c, x) \tag{14}$$

and

$$y^1 \text{ minimizes } (b, y).$$

Our present interest is where (5) is (L_1), and (6) is (L_2) and (L_3). For any linear function $(c, x)={}_j\Sigma c_j x_j$ of points $x\epsilon P_G$, we get a dual system (7), (8), (b, y), by letting a variable y_ξ correspond to each inequality of L_2 and L_3. That is let a variable y_h correspond to each node $\nu_h\epsilon G$ and let a variable y_s correspond to each set S of two or more nodes in G.

For (7) we have,

$$\text{for every } \nu_h,\ y_h \geqslant o, \tag{15}$$

and

$$\text{for every } S,\ y_s \geqslant 0. \tag{16}$$

Coefficient $a_{hj}=1$ if edge e_j is directed toward node ν_h, and $a_{hj}=0$ otherwise. Coefficient $a_{sj}=1$ if edge e_j has both ends in S, and $a_{sj}=0$ otherwise.

For every v_h, $b_h=1$. For every S, $b_s=|S|-1$.

Therefore, (8) becomes

> for every edge $e_j \epsilon G$,
> $y_h+w_j \geq c_j$, where v_h is the front end
> of e_j, and where $w_j=\Sigma y_s$, summed over
> all sets S which contain both ends of e_j. (17)

Function (b, y) becomes

$$(b, y)={}_h\Sigma y_h+{}_s\Sigma(|S|-1)y_s,$$

summed over all v_h and over all S.

Recall that our task is to show that the vector x^0 of the branching B^0, produced by the algorithm, maximizes (c, x) over P_G.

In view of (14), we do so by constructing a vector $y=[y_h, y_s]$ which satisfies (15), (16), (17), and which satisfies (12) and (13). For the present system, (12) is

> for every node v_h such that $y_h \neq 0$,
> $\Sigma x_j^0=1$, summed over j's such that e_j
> is directed toward v_h; (18)

and

> for every set S such that $y_s \neq 0$,
> $\Sigma x_j^0=|S|-1$, summed over j's
> such that e_j has both ends in S. (19)

In other words (18) says that if $y_h \neq 0$ then an edge of the branching B^0 is directed toward v_h, and (19) says that if $y_s \neq 0$ then exactly $|S|-1$ edges of B^0 have both ends in S.

For the present system (13) is

> for every edge e_j in the branching B^0,
> $y_h+w_j=c_j$, where v_h and w_j are as in (17). (20)

Section 7

For each graph $G^i(i=k, k-1, \ldots, 0)$ with weight c_j^i on each edge $e_j^i \epsilon G^i$, and for the branching B^i in G^i, we will describe a vector y^i which satisfies (15)–(20), where G and B^0 are replaced by G^i and B^i and where vector y is y^i.

First we describe a y^k, and then, assuming a

$$y^{i+1}(i=k-1, \ldots, 0),$$

we describe a y^i. Thus by induction we obtain a $y=y^0$ and the proof of Theorems 1 and 2.

The vector $y^k=[y_h^k, y_s^k]$ is $y_s^k=0$ for every set S of two or more nodes in G^k, $y_h^k=0$ for every node v_h^k in G^k which has no edge of B^k directed toward it, and, for every other node v_h^k in G^k, $y_h^k=c_j^k$ where edge e_j^k of B^k is directed toward v_h^k. Conditions (15)–(20) for y^k can be immediately verified from the fact that for every node $v_h^k \epsilon G^k$ either there is no edge of B^k directed toward v_h^k and there is no positively weighted edge directed toward v_h^k, or else, among all the positively weighted edges directed toward v_h^k, the one in B^k has maximum weight.

Now, suppose that we have a y_h^{i+1} for each node v_h^{i+1} and a y_s^{i+1} for each set S of two or more nodes in G^{i+1}, such that (15)–(20) are satisfied (where B^0 is replaced by B^{i+1}, etc.).

Let $t_h^{i+1}=\Sigma y_s^{i+1}$, summed over the sets S which contain node v_h^{i+1}.

To make the induction go through we assume further that in G^{i+1}

> for every node v_h, such that $t_h+y_h>0$,
> there exists at least one edge e_j directed
> toward v_h such that $c_j=t_h+y_h$. (21)

This clearly holds for G^k, and we will prove from (15)–(21) for G^{i+1} that (15)–(21) holds for G_i.

Obtain the vector y^i as follows:

Where A is the set of nodes in circuit Q^i of G^i, where e_2^i is the edge of Q^i not in B^i, where v_2^i is the front end of e_2^i, where c_0^i is the minimum weight in Q^i, and where v_1^{i+1} is the node in G^{i+1} to which Q^i was shrunk, let

$$y_2^i=y_1^{i+1}+c_2^i-c_0^i, \tag{22}$$

and

$$y_A^i=c_2^i-y_2^i-t_1^{i+1}. \tag{23}$$

Where v_3^i is any node in A other than v_2^i, and where e_4^i is the edge in Q^i which is directed toward v_3^i, let

$$y_3^i=c_4^i-y_A^i-t_1^{i+1}. \tag{24}$$

Observe that (24) holds also for $v_3^i=v_2^i$.

Where v_5^i is any node of G^i which is not in Q^i, let

$$y_5^i=y_5^{i+1}. \tag{25}$$

Where R is a nonempty subset of nodes in G^{i+1} which does not contain v_1^{i+1}, where $J=R \cup v_1^{i+1}$, where $K=R \cup A$, and where L is any set of two or more nodes in G^i such that $L \cap A$ is a proper subset of A, let

$$y_K^i=y_R^{i+1}, \tag{26}$$

$$y_K^i=y_J^{i+1}, \tag{27}$$

and

$$y_L^i=0. \tag{28}$$

That completes the description of vector y^i. Now we must verify (15)–(21) for it.

For every edge of G^i which is directed toward a node not in A, for every node not in A, and for every set S, except A, in G^i, conditions (15)–(18), (20), and (21) follow immediately from those same conditions for y^{i+1}, (25)–(28), and the local nature of the change from G^{i+1}, B^{i+1}, and c^{i+1} to G^i, B^i, and c^i.

For every subset of nodes in G^i which does not contain all of A, condition (19) follows immediately as above. For set A and for every set K as in (27), condition (19) follows from (27), condition (19) for set J in G^{i+1}, and the fact that there are exactly

$$|K| - |J| = |A| - 1$$

more edges of B^i with both ends in K than there are edges of B^{i+1} with both ends in J, namely the edges of $B^i \cap Q^i$.

It follows from (24), (27), and (28), that (21) holds for every node ν_3^i in A (in particular where e_j is the e_4^i of (24)), and that (20) holds for every edge of $B^i \cap Q^i$, and that (17) holds for e_2^i.

Condition (18) follows immediately for each node of A except ν_2^i since there is an edge of $B^i \cap Q^i$ directed toward it. If there is an edge e_1^{i+1} in B^{i+1} which is directed toward ν_1^{i+1}, then e_1^i is an edge of B^i which is directed toward ν_2^i, and so in this case (18) follows for ν_2^i. Otherwise, if there is no edge of B^{i+1} directed toward ν_1^{i+1}, then by (18) for ν_1^{i+1}, $y_1^{i+1} = 0$. Also in this case, the c_2^i of (22) was chosen in the algorithm to be c_0^i. Therefore, if there is no edge of B^{i+1} directed toward ν_1^{i+1}, then (22) is $y_2^i = 0$, and so (18) follows for ν_2^i.

For e_1^i, the only edge, if any, which is in $B^i - Q^i$ and directed toward a node in A, we have

$$c_1^{i+1} = c_1^i + c_0^i - c_2^i \text{ (from (1))}, \quad (22), \quad y_1^{i+1} + w_1^{i+1} = c_1^{i+1}$$

which is (20) for e_1^{i+1}, and $w_1^i = w_1^{i+1}$ from (27) and (28). Combining these we get $y_2^i + w_1^i = c_1^i$, which is (20) for e_1^i.

Thus conditions (18), (19), (20), and (21) are now completely accounted for. Condition (17) for edges not in Q^i but directed toward nodes in A, condition (16) for y_A^i, and condition (15) for nodes in A, remain to be verified.

Let e_5^i be any edge of G^i which has both ends in A, and let ν_3^i be its front end. To prove (17) for e_5^i, which is $y_3^i + w_5^i \geq c_5^i$ where $w_5^i = y_A^i + t_1^{i+1}$, combine (24) and $c_4^i \geq c_5^i$.

Let e_3^i be any edge of G^i which has its front end ν_3^i in A and its rear end not in A. To prove condition (17) for e_3^i, which is $y_3^i + w_3^i \geq c_3^i$ where $w_3^i = w_3^{i+1}$, combine (24), (23), (22), (1), and (17) for e_3^{i+1}.

To prove (16) for A, that is $y_A \geq 0$, we use (21) for ν_1^{i+1}. Assuming $t_1^{i+1} + y_1^{i+1} > 0$, let e_3^{i+1} be the e_j of that relation, let ν_3^i be the front end of e_3^i in A, and let e_4^i be the edge of Q^i which is directed toward ν_3^i. Here (21) is $c_3^{i+1} = t_1^{i+1} + y_1^{i+1}$. In this case, obtain $y_A \geq 0$ by combining (23), (22), (21) for ν_1^{i+1}, (1), and (4).

If there is no e_3^{i+1} directed toward ν_1^{i+1} such that $c_3^{i+1} = t_1^{i+1} + y_1^{i+1}$, then $t_1^{i+1} + y_1^{i+1} = 0$, and all edges directed toward ν_1^{i+1} have negative weight in G^{i+1}, so none of them are in B^{i+1}. Therefore since in this case the c_2^i of (22) was chosen to be c_0^i, (22) becomes $y_2^i = 0$, and (23) becomes $y_A^i = c_0^i$. By (2), we have $y_A^i \geq 0$.

Prove (15) for any node ν_3^i in A by combining (24), (23), (22), (3), and $y_1^{i+1} \geq 0$.

That completes the proof of Theorems 1 and 2.

Section 8

Notice from the proof that if every weight c_j, $e_j \epsilon G$, is an integer, then the vector y^0, as well as vector x^0, is integer-valued. In particular, where every $c_j = 1$, vector y^0 is 0,1-valued and $\max(c, x) = \min(b, y)$ is a simple "Konig-type" theorem, analogous to the maximum-cardinality-matching duality theorem in [1].

The following two theorems can be proved by the methods used here.

THEOREM 3. *Where* (L_4) *is* $\Sigma x_j = n$, *summed over all edges* $e_j \epsilon G$, *the vertices of the polyhedron given by* (L_1), (L_2), (L_3), *and* (L_4) *are precisely the vectors of the* n*-cardinality subsets of edges in* G *which comprise branchings. (In particular, where* n *is one less than the number of nodes in* G, *these branchings are the spanning arborescences of* G).

The present research began when A. J. Goldman asked for a description of "the convex hull of the spanning trees of a graph." Theorem 4 is proved in [3].

THEOREM 4. *The vertices of the polyhedron* F_G *given by* (L_1) *and* (L_3) *are precisely the vectors of the subsets of edges in* G *which comprise forests. The vertices of the intersection of* F_G *with* (L_4) *are a subset of the vertices of* F_G.

Section 9

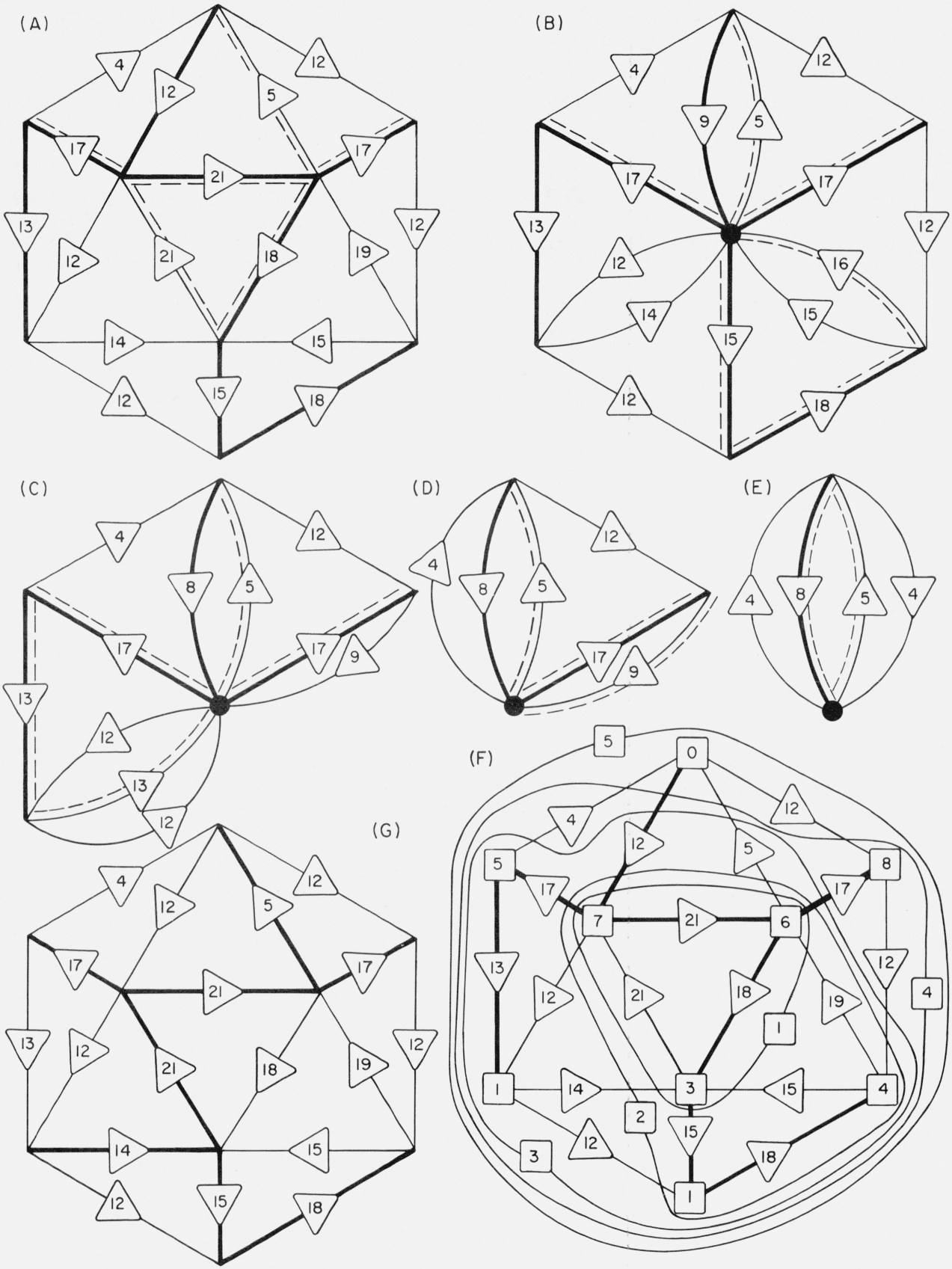
Figures (A) through (E) illustrate the algorithm for finding an optimum (i.e., maximum total weight) branching in graph (A). Each dashed edge is dashed because it has maximum positive weight among those edges directed toward its front end. As soon as a dashed circuit arises it is shrunk and certain new edge weights are computed, thereby producing the edge-weighted graph of the next figure. The final graph is not drawn since it is simply a node. After the sequence of figures is completed, except for the boldness of the bold edges, then working backwards through the sequence, appropriate dashed edges are made bold. The answer is the branching in (A) formed by the bold edges. It is the only correct answer.

Figure (F) illustrates a "dual answer," y, for the same problem. The numbers in the squares on the nodes are the values of the nodal y-variables. The numbers in the squares on the closed curves are the nonzero values of y-variables corresponding to subsets of nodes. Each closed curve encloses the subset of nodes to which its number corresponds. Observe that the vector y, thus represented, satisfies relations (15)–(20), and thus guarantees that the branching is optimum. This y is not the one described in section 8.

The example (A)–(E) was actually obtained by constructing it all except the edge-weights first. Then all the numbers in (F) were chosen so as to yield the structure (A)–(E).

Figure (G), by coincidence, nicely represents three different things.

It illustrates a nonoptimum branching obtained by applying the greedy algorithm. This branching has total weight 128, whereas the branching shown in (A)



FIGURES A–G

has total weight 131. One other branching that might be obtained by applying the greedy algorithm has total weight 127.

Since the optimum branching shown in (A) is a spanning arborescence, it is an optimum spanning arborescence, and thus it is still an optimum spanning arborescence when all of the edge-weights are changed by adding any constant. However, when all the edge-weights are changed by adding the constant, -10, the branching shown in (A) is no longer an optimum branching, even though all the weights in this branching are still positive, and even though this branching is still an optimum spanning arborescence. For these new edge-weights there are two optimum branchings, quite different from each other, and both quite different from the branching shown in (A). Neither one of them is a spanning arborescence. One of them consists of all the bold edges in (G) except for the edge which is weighted -5 (relative to the new weights). This branching has total weight, 53, whereas the branching shown in (A) has total weight, 51.

A direct algorithm for finding, if there is one, an optimum spanning arborescence is obtained from the algorithm for finding an optimum branching simply by deleting the words "positively weighted" from (I1). This follows from the fact that the only effect on the resulting algorithm of adding a constant to each number of an input is to add the same constant to every number that arises in the algorithm.

Because of the words "positively weighted" in the optimum branching algorithm, the effect of adding -10 to each edge-weight in (A) is that the resulting application of the optimum branching algorithm does not dash the edge that is weighted -5, and does not dash in (D) the edge that is weighted -1. Thus there is no shrinking in (D); the sequence of graphs stops at (D). In (D), only the edge that is weighted 7 is made bold since it is the only one that gets dashed. (The present computation is not explicitly illustrated.) Unlike in the spanning arborescence problem, we have in (C) no bold edge directed toward a node in the dashed circuit. Therefore, the edges of the dashed circuit in (C), except for one or the other of its minimum-weight edges, are made bold. The choice here is what gives rise to the two correct answers. It is interesting to note that the two edges which are tied in this step of the computation do not have the same edge-weights in (A), and that the two optimum branchings, arising from the two choices, are globally quite different. I recommend carrying through the completion of each.

Figure (G) also illustrates an optimum spanning arborescence having the lower left node prescribed as root. The first phase of the algorithm for obtaining it is the dashing and shrinking and computing of new edge weights just as in (A)–(E). The only difference from (A)–(E) is the way the edges are chosen from among the dashed ones to be made bold. An optimum spanning arborescence rooted at any other prescribed node is obtained from this same first phase of (A)–(E) by appropriately choosing edges from among the dashed ones. The subgraph of (A), formed by the image in (A) of all edges dashed somewhere in (A)–(E), in general contains nonoptimum spanning arborescences with prescribed root as well as optimum ones. Therefore, the choosing does depend on the structure of the sequence (A)–(E). In general, it is a nice feature of the computations for finding in the same edge-weighted graph, when they exist, optimum spanning arborescenses with various prescribed roots, that these computations are identical except for the (I3) part.

This paper was to have appeared in the published proceedings of the International Seminar on Graph Theory and Its Applications, Rome, July 1966, sponsored by the International Computation Center. Various international failures of communication during the editorial process precluded it. I am sorry to have lost that opportunity to record my contribution to an outstanding symposium. I wish to acknowledge here my appreciation to the organizers of the symposium for the excellent job they did and for their kindness to me.

References

[1] Jack Edmonds, Paths, trees, and flowers, Canadian J. Math. **17**, 449–467 (1965).

[2] Jack Edmonds, Maximum matching and a polyhedron with 0,1-vertices, J. Res. NBS **69B** (Math. and Math. Phys), Nos. 1 & 2, 125–130 (1965).

[3] Jack Edmonds, Matroids and the greedy algorithm, to appear.

[4] R. E. Gomory, The traveling salesman problem, Proceedings of the IBM Scientific Computing Symposium on Combinatorial Problems, 1966, pp. 93–117.

[5] J. B. Kruskal, On the shortest spanning subtree of a graph and the traveling salesman problem, Proc. Amer. Math. Soc. **7**, 48–50 (1956).

[6] P. Rosenstiehl, L'arbre minimum d'un graphe, International Seminar on Graph Theory, Rome, July 1966.

(Paper 71B4–249)