JOURNAL OF RESEARCH of the National Bureau of Standards—B. Mathematics and Mathematical Physics
Vol. 69B, Nos. 1 and 2, January–June 1965

Maximum Matching and a Polyhedron With 0,1-Vertices[1]

Jack Edmonds

(December 1, 1964)

A matching in a graph G is a subset of edges in G such that no two meet the same node in G. The convex polyhedron C is characterized, where the extreme points of C correspond to the matchings in G. Where each edge of G carries a real numerical weight, an efficient algorithm is described for finding a matching in G with maximum weight-sum.

Section 1

An algorithm is described for optimally pairing a finite set of objects. That is, given a real numerical weight for each unordered pair of objects in a set Y, to select a family of mutually disjoint pairs the sum of whose weights is maximum. The well-known optimum assignment problem [5][2] is the special case where Y partitions into two sets A and B such that pairs contained in A and pairs contained in B are not positively weighted and therefore are superfluous to the problem. For this "bipartite" case the algorithm becomes a variant of the Hungarian method [3].

The problem is treated in terms of a graph G whose nodes (vertices) are the objects Y and whose edges are pairs of objects, including at least all of the positively weighted pairs. A *matching* in G is a subset of its edges such that no two meet the same node in in G. The problem is to find a maximum-weight-sum matching in G. The special case where all the positive weights are one is treated in detail in [2] and [6]. The description here of the more general algorithm uses the terminology set up in [2]. Paper [2] (especially sec. 5) helps also to motivate this paper, though it is not really a prerequisate till section 7 here.

The increase in difficulty of the maximum weight-sum matching algorithm relative to the size of the graph is not exponential, and only moderately algebraic. The algorithm does not involve any "blind-alley programming"—which, essentially, amounts to testing a great many combinations.

The emphasis in this paper is on relating the matching problem to the theory of continuous linear inequalities. In particular, we prove a theorem analogous to one of G. Birkhoff [1] and J. von Neuman [5] which says that the extreme points of the convex set of doubly stochastic matrices (order n by n) are the permutation matrices (order n by n). That theorem and the Hungarian method are based on Konig's theorem about matchings in bipartite graphs. Our work is related to results on graphs due to Tutte [4].

There is an extensive related literature besides these references. One may refer to surveys on graphs, linear programming, network flow, and combinatorial analysis (other than enumeration). However, paper [2] and this one are together self-contained. For the algorithm without the polyhedral geometry, sections 4 and 7, here and in [2], suffice.

The technique, described in sections 2 and 3, of using linear programming duality to derive a description of the convex polyhedra associated with a class of combinatorial structures appears applicable, whereever the combinatorics is adequately understood, independently of the particular nature of the associated algorithm. The results of this paper suggest that, in applying linear programming to a combinatorial problem, the number of relevant inequalities is not important but their combinatorial structure is.

In another paper, I will extend the present work to "Optimum degree-constrained subgraphs". See the end of this paper for two main results of that extension.

Section 2

Let the real variables $x \epsilon E$ correspond to the edges e of a finite graph G. Let C be the convex polyhedron of vectors $< x >$ formed by the intersection of all the half-spaces given by the following inequalities (1), (2), and (3).

[1] Prepared while the author was a research associate at Princeton University, under the Princeton Logistic Research Project (Office of Naval Research). Based on work with the NBS Combinatorial Mathematics Project (Army Research Office, Durham).

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

(1) $x \geq 0$, for all $x \epsilon E$.

(2) for every node v of G, $\Sigma x \leq 1$ (summed over $x \epsilon V$), where V is the set of variables corresponding to the edges of G which meet node v.

(3) for every subset S of $2r+1$ nodes in G where r is a strictly positive integer, $\Sigma x \leq r$ (summed over $x \epsilon R$), where R is the set of variables corresponding to the edges of G with both ends in S.

Condition (I) is that the variables x take on only values zero and one. Assuming condition (I), each vector $< x >$ represents and may be regarded as equivalent to the subset of edges in G which correspond to the one-valued components of $< x >$.

Assuming (I), condition (2) is the definition of a matching in G. Assuming (I), condition (3) says that for any set S of $2r+1$ nodes (r, a positive integer) the set $< x >$ of edges contains no more than r edges with both endpoints in S; clearly this is implied by the set of edges being a matching in G. Therefore, assuming (I), condition (2) implies condition (3) for vectors $< x >$. However, replacing (I) by the weaker condition (1), it is easy to show that, where G contains a circuit with an odd number of edges, condition (2) does not imply condition (3) for vectors $< x >$.

The essence of our following theorem (P) is that for purposes of linear-extremizing over the family of matchings in G, discreteness condition (I) and condition (2) can be replaced by the inequalities (1), (2), and (3).

Let P be the set of vectors $< x >$ such that each component is a zero or one and such that the one-components correspond to the edges of a matching in G. That is let P be the vectors $< x >$ which satisfy (I) and (2). We call $< x > \epsilon P$ a matching vector of G.

THEOREM (P): P *is the set of vertices (extreme points) of polyhedron* C.

Unless the graph G has an edge joining each pair of nodes, inequalities (1), (2), and (3) generally include more than the minimal set of bounding planes for C, but that is not so important. What is important in order to provide a good characterization for maximum weight-sum matchings in a graph G is a good characterization of *some* family of inequalities which together bound precisely the convex hull of P.

It is obvious that the points P are vertices of C—that is they belong to C and none lie half way between two other points of C. In fact, they are vertices of the larger polyhedron, C', given by inequalities (1) and (2), and they are not sliced away by (3). However, in C' there are other vertices. What remains to be proved is that vertices P are the *only* vertices in C.

Every linear form in the variables of the space of a convex polyhedron is maximized over points in the polyhedron, if a maximum exists, by a vertex (and perhaps other points as well). Conversely, every vertex is the unique maximum over the polyhedron of some linear form.

In particular, where each edge e of G carries a real weight c, theorem (P) implies that the maximum weight-sum for matching in G equals the maximum of

$$(4)\quad W = \Sigma c x \text{ (summed over } x \epsilon E),$$

where real vector $< x >$ satisfies (1), (2), and (3). And conversely, theorem (P) follows by proving that for all real $< c >$, W is maximized by a vector $< x >$ whose components are zeroes and ones.

Section 3

Where $[a_{ij}]$ is any real matrix and $< b_j >$ and $< c_i >$ are real vectors, let $< x_i >$ represent the vectors satisfying the inequalities $x_i \geq 0$ *and* $\Sigma a_{ij} x_i \leq b_j$ (summed over i) and let $< y_j >$ represent the vectors satisfying the inequalities $y_j \geq 0$ and $\Sigma a_{ij} y_j \geq c_i$ (summed over j). The duality theorem of linear programming says that $\max \Sigma c_i x_i = \min \Sigma b_j y_j$ when these extrema exist for vectors $< x_i >$ and $< y_i >$.

To get the linear program which is dual to maximizing W of (4) in the polyhedron C, we introduce a new variable corresponding to each inequality of (2) and (3). That is for each node v in G we introduce a variable, call it y, and for each set S of $2r+1$ nodes in G (r, a positive integer) we introduce a variable, call it z.

Let $< y, z >$ denote the vector of all variables y and z. The duality theorem says that the maximum of W is equal to the minimum of

(5) $U = \Sigma y + \Sigma r z$ (first sum taken over all nodes v and second sum taken over all sets S), where vector $< y, z >$ satisfies the following nonnegativity inequalities (6), and the following inequalities (7), obtained from the transpose of the matrix of coefficients of inequalities (2) and (3).

(6) For every node v, $y \geq 0$; for every set S, $z \geq 0$.

(7) For every edge e of G and the nodes v_1 and v_2 which it meets, $y_1 + y_2 + \Sigma z \geq c$ (where y_1 and y_2 are the variables corresponding to v_1 and v_2 and where the sum is taken over all z for which the corresponding set S contains v_1 and v_2). The c is the coefficient in linear form W of the x which corresponds to edge e. In other words, c is the weight on edge e.

For a fair sized graph G, vector $< y, z >$ has a huge number of components. However, in general any vector which is a vertex of a dual polyhedron has no more nonzero components than the total number of variables in the primal linear program. In particular, any vector $< y, z >$ with which we deal will have no more nonzero components than the number of edges in G.

As easily shown in general for dual linear programs, $W \leq U$ for all admissible vectors $< x >$ and $< y, z >$. Therefore to prove that any linear form W is maximized in C by some matching vector, that is to prove theorem (P), it is sufficient to display a vector $< x > \epsilon P$ and some vector $< y, z >$ satisfying (6) and (7) such that $W = U$. Notice that we are using here only the easy part of the duality theorem. In fact, by displaying the vectors as described we will be proving, for the class of programs given by C and W, the harder part of the duality theorem, though that is incidental.

Conversely, theorem (P) and the duality theorem (including the harder part) imply that a matching M in G has maximum weight-sum if and only if that weight-sum equals U for some $< y, z >$ (with a mod-

erate number of nonzero components) which satisfies (6) and (7).

For any coefficients $< c >$ for W, that is for any weights $< c >$ on the edges of G, we shall describe a matching vector $< x >_0 \epsilon P$, corresponding to a matching M in G, and a vector $< y, z > = < y, z >_0$ such that (6) and (7) hold and furthermore such that the following (8), (9), and (10) hold.

(8) $y = 0$ for each node v which is not an endpoint of an edge in M.

(9) equality holds for each instance of (7) where $e \epsilon M$.

(10) for each $z > 0$, the set S contains both endpoints of r edges in M.

Therefore, summing together on each side the equations of (8) and (9), we will get an instance of $U = W$, where $< y, z > = < y, z >_0$ and $< x > = < x >_0$.

Section 4

We turn now to theorem (M) which, for any matching M corresponding to a matching vector $< x >_0$ which maximizes $W = cx$, will yield a vector $< y, z >_0$, satisfying (6), (7), (8), (9), and (10). From this and from the discussion in section 3, it follows that the existence of a vector $< y, z >_0$ is a necessary and sufficient condition for a matching M to have maximum weight-sum. Theorem (M) itself is also such a condition in a different form. Theorem (M) displays the existence of much tighter and more complex structure than simply the vector $< y, z >_0$, tighter and more complex than really necessary for characterizing maximum matchings.

Ellis Johnson and Charles Zahn in studying this theory found that theorem (M) can be bypassed in obtaining the vector $< y, z >_0$ and that the maximum matching algorithm can be executed in terms of parameters of type y and z rather than the numerical parameters of theorem (M). One type of parameter may be arithmetically more convenient than the other, though the same combinatorial manipulations seem essential in either case. Theorem (M) seems justified by the insight it provides and by its natural relationship to the combinatorial manipulations of the algorithm, so it might as well be proved on the way to proving theorem (P). It is a direct consequence of the algorithm as described in section 7. More important, it is part of the description.

THEOREM (M): *For graph* G *with edge-weights* $< c >$, *a matching* M *is maximum if and only if there exists a sequence* $\{G_i\}$ $i = 0, \ldots, n$.

Each G_i is a graph together with a matching M_i, a numerical weight $w(e^i)$ for each edge $e^i \epsilon G_i$, and a numerical weight $w(v^i)$ for each node $v^i \epsilon G_i$. Sequence $\{G_i\}$ has the following properties.

(a) G_0 is graph G with edge-weights $w(e) = c$, with matching $M_0 = M$, and with any vertex-weights satisfying the general conditions below.

(b) $w(v^i) \geq 0$ for all $v^i \epsilon G_i$ and $w(v_1^i) + w(v_2^i) \geq w(e^i)$ for all $e^i \epsilon G_i$ where v_1^i and v_2^i are the endpoints of e^i.

(c) For $i = 0, \ldots, n-1$, there is in G_i a circuit (simple closed path) B_i containing $2a_i + 1$ edges, a_i of them in M_i. Circuit B_i is called a *blossom* in (G_i, M_i).

(d) $w(v_1^i) + w(v_2^i) = w(e^i)$ for $e^i \epsilon B_i$.

(e) If a vertex of B_i meets no edge of M_i, it will be one of the vertices, say q^i, with smallest weight, $w(q^i)$, in B_i.

(f) To obtain graph G_{i+1} from graph G_i shrink B_i and all edges with both end-points in B_i to a single node u^{i+1} in G_{i+1} which, in place of the vertices of B_i, is the new endpoint of those edges having with respect to G_i one endpoint in B_i. The matching in G_{i+1} is $M_{i+1} = M_i \cap G_{i+1}$.

(g) Weights in G_{i+1} are the same as corresponding weights in G_i except at u^{i+1} and edges meeting u^{i+1}.

(h) For minimum $w(q^i)$ in B_i, $w(u^{i+1}) \leq w(q^i)$.

(i) For each edge e^{i+1} meeting u^{i+1}, let e^i be the corresponding edge in G_i, meeting v^i of B_i. Then

$$w(e^{i+1}) = w(e^i) - w(v^i) + w(q^i).$$

(j) For G_n, $w(v_1^n) + w(v_2^n) = w(e^n)$ for all $e^n \epsilon M_n$.

(k) For a vertex $v^n \epsilon G_n$ not meeting an edge in M_n, $w(v^n) = 0$.

Section 5

Vector $< y, z >_0$ is as follows. The node-sets S for which the corresponding z in $< y, z >_0$ is positive are among those, say S_i, corresponding to the B_i of theorem (M). The nodes in each S_i are the nodes of G which have been absorbed into the nodes of B_i (i.e., into u^{i+1}) in the process of going from graph $G = G_0$ to graph G_i. Define

(11) $d_i = w(q^i) - w(u^{i+1})$ for each S_i, and $d = 0$ for every other set S.

(12) $z = 2d$, for all S.

For each y component of $< y, z >_0$, corresponding to node v of G, we set

(13) $y = w(v) - \Sigma d$ (summed over d corresponding to sets S for which $v \epsilon S$), where $w(v)$ is the weight assigned by theorem (M) to node v in $G_0 = G$.

PROOF: For each edge $e = e^\circ$ with endpoints v_1 and v_2 in $G = G_0$, let e^j with endpoints v_1^j and v_2^j be the corresponding edge in G^j, where j is such that either $j = n$ or else e^j has both ends in B_j and thus is absorbed into u^{j+1}.

By virtue of (g), in the applications of formula (i) to e^j and its pre-images, each $w(v^j)$ term is either $w(v_1)$ or $w(v_2)$ or else a certain $w(u^{k+1})$ where u^{k+1}, the contraction of a B_k, absorbs one but not both of v_1 and v_2 in going from G_0 to G_n. Conversely, each such $w(u^{k+1})$ is either $w(v_1^j)$ or $w(v_2^j)$ or else a certain $w(v^j)$ term in one of the applications of (i) to e^j and its pre-images. Furthermore, $w(v_1)$ is a certain $w(v^j)$ term if and only if $w(v_1^j)$ is a certain $w(u^{k+1})$. Otherwise, $w(v_1) = w(v_1^j)$. Similarly for $w(v_2)$. Thus by virtue of (g), repeated application of (i) and then repeated substitution of (11), yields:

(14) $w(e^j) = w(e) + w(v_1^j) + w(v_2^j) - w(v_1) - w(v_2) + \Sigma d_k$, where k is summed over all the B_k (or u^{k+1}) into which either v_1 or v_2, but not both, are absorbed in going from G_0 to G_n.

757-615 O-65-9

By (b) and (14),

(15) $w(v_1)+w(v_2)-\Sigma d_k \geq w(e)=c$, where k is summed as in (14). Substituting (13) twice and (12) several times we get (7).

By (d), (j), and (14), equality holds for (15) and thus for (7) when $e \epsilon M$. Hence (9) is verified.

Substituting (11) for nonzero d in (13), recollecting terms, and using (g), we have

(16) $y=w(v^n)+\Sigma[w(v^k)-w(q^k)]$, where k is summed over all the B_k into which v is absorbed and where v^k and v^n are the images of v in B_k and G_n. Hence, by (b) and (e), we have $y \geq 0$. By (h) and (11), $z \geq 0$. Thus, (6) is verified.

When a node v meets no edge of M, by (c) and (f) neither does any image v^i meet an edge of M_i. Therefore by (e) and (k) and (16), $y=0$ in this case. Thus, (8) is verified.

Condition (10) follows from (c) and (f). Thus, modulo the "only if" part of theorem (M), conditions (6) through (10) are verified for some $< y, z >_0$ and theorem (P) is proved.

Section 6

Since $U \geq W$ in general, the "if" part of theorem (M) is proved by the above translation from $\{G_i\}$ to a $< y, z >_0$ for which $U=W$.

It is much simpler for the matching algorithm, given $< c >$, to construct *some* maximum matching with a $\{G_i\}$ than to construct a $\{G_i\}$ for a particular maximum matching. Indeed, the simpler construction is all that is needed to get theorem (P). However, by a continuity argument we can show the existence of a $\{G_i\}$ for any maximum matching M and thereby complete the proof of theorem (M).

Add a positive ϵ to the weight c of each edge e in M; leave all other edge weights the same. With this new $< c >$, M is the unique maximum matching and therefore, by the simpler construction, the algorithm in section 7, there exists a $\{G_i\}$ for it.

An infinite sequence of sequences $\{G_i\}$, corresponding to an infinite sequence of ϵ's which approach zero, contains only a finite number of different combinatorial configurations aside from the values of the weights. The space of all possible weights is bounded and finite-dimensional. Hence there is a subsequence of $\{G_i\}$'s, combinatorially the same, with a limit which is a $\{G_i\}$ like we want.

Section 7

The maximum-weight-sum matching algorithm consists of the maximum cardinality algorithm in [2] (secs. 4 and 7) together with small modifications suggested by theorem (M).

Suppose we have a sequence $\{G_i\}$ ($i=0, \ldots, m$), with not-necessarily-maximum matchings $\{M_i\}$, satisfying all the conditions (a) through (j), omitting only (k). To get a weak sequence of this kind to start with—take $m=0$, take the matching to be empty, and take sufficiently large vertex weights.

We apply the algorithm in [2] to matching M_m in the subgraph, G'_m, of G_m which consists of all nodes of G_m plus all edges e^m for which $w(v_1^m)+w(v_2^m)=w(e^m)$. If G_m does not satisfy (k) then there is an exposed node r in G'_m, for which $w(r)>0$. (*Exposed* means it meets no matching edge.) Start growing in G'_m a planted tree rooted at r.

If it grows into a flowered tree with blossom B_m, then in G_m shrink B_m to a u^{m+1} to obtain a G_{m+1}. Where q_m is the node in B_m with smallest weight, set $w(u^{m+1})$ equal to $w(q^m)$ and adjust the weights of the edges which meet u^{m+1} according to the formula in (i). Leave other weights in G_{m+1} the same as in G_m. Thus, weights in G_{m+1} will satisfy conditions (b), (g), (h), (i), and (j). Furthermore, G'_{m+1} is the image in G_{m+1} of G'_m, and so we continue in G'_{m+1} with the tree image.

Eventually, possibly after a number of shrinkings, we obtain either: (1) an augumenting tree or, (2) a tree which has an outer vertex v with $w(v)=0$ or, (3) a Hungarian tree which has outer vertices all with positive w.

In case 1, augment. That is, interchange matching roles of edges in the augmenting path. This yields a matching with larger weight-sum in the graph, still call it G_m, and disposes of one or two vertices violating (k). In case 2, the path in the tree joining v to r is really "augmenting" also, though such paths are not encountered in [2]. Treat it like the path in case 1 and it serves the same purpose.

The fact that a matching is obtained which has larger weight-sum is not needed for the validation of the algorithm. The important fact is the decrease in the number of nodes r in the final term G_m of $\{G_i\}$, such that r is exposed and $w(r)>0$. After an augmentation in case 1 or 2, abandon the tree and start a new tree at another node r if there is one.

When we get, in case 1 and 2, a matching with larger weight-sum in G_m, it yields a matching with larger weight-sum in each graph G_i back through G_0. It does this by the process of successively selecting a new matching within each blossom B_i which is compatible in G_i with the matching already chosen with respect to G_{i+1}. If u^{i+1}, the blossom B_i before "expansion", is an exposed node in G_{i+1} then the selection of matching edges in B_i is determined by condition (e). Sequence $\{G_i\}$ with these new matchings M_i satisfies (a) through (j) and comes closer by at least one to satisfying (k). There is no advantage in selecting the matching in graphs G_i other than G_m until the algorithm is otherwise finished or until it is necessitated by condition (h) in case (3).

It is important for the type of step of the algorithm necessitated by (h) in case 3, described below, that we be able to "expand" the shrunken blossoms B_i in an order different from the order in which they were shrunk—thereby obtaining a sequence $\{G_i^a\}$ of type $\{G_i\}$ (possibly not satisfying (k)) such that the terms G_0^a and G_m^a together with their weights and matchings are, respectively, identical to the terms G_0 and G_m of $\{G_i\}$ and such that the blossoms B_i^a of $\{G_i^a\}$ correspond in a different order to the blossoms B_i of $\{G_i\}$. The order of the blossoms B_i is not arbitrary but it is limited only by the relation of one u^{i+1} having been absorbed into another.

In particular, for any node v^m of G_m which is the image of some B_i, let B_k be the blossom of $\{G_i\}$ whose image in G_m is v^m and such that k is maximum. Node v^m can be "expanded" to obtain from G_m with its weights and matching a certain graph, G_{mk}, with weights and a matching and a blossom, B_{mk}, corresponding to B_k. The graph structure, the weights, and the matching of G_{mk} in the neighborhood of B_{mk} are determined by G_m in the neighborhood of v^m and by G_k in the neighborhood of B_k, subject to the conditions (other than (k)) of theorem (M). Elsewhere G_{mk} is like G_m. There exists a sequence $\{G_i^a\}$ as described above such that the next to the last term is G_{mk}. Without describing it and justifying it here in any greater detail, the derivation of $\{G_i^a\}$ should be fairly evident from the local nature of the changes in successive terms of any sequence $\{G_i\}$.

In implementing the algorithm it may be better to represent the partial order structure of inclusion for the blossoms B_i of a $\{G_i\}$ so that no special preference is given to $\{G_i\}$ or any of the other sequences $\{G_i^a\}$. Here we describe the algorithm with respect to a particular $\{G_i\}$ merely for convenience of exposition.

In case 3, the weights of the vertices in the Hungarian tree are adjusted. Weights of outer vertices are lowered and weights of inner vertices are raised by a uniform amount, which is as large as possible without violating for the sequence $\{G_i\}$ either (b) by making an outer vertex too small or (h) by making an inner vertex which is the image of a shrunken blossom too large. In the adjustment, property $w(v_1^m)+w(v_2^m)=w(e^m)$ is preserved for edges of the tree. Edges of G'_m meeting an inner vertex and not an outer vertex will drop out of G'_m.

If (b) limits the adjustment, it may be because some outer vertex weight becomes zero. Then, possibly after the operation performed in case 2, we again have one less node violate (k). Otherwise if (b) limits the adjustment, it is because for one or more edges e^m in G_m and not in G'_m, which meet outer vertices, we get $w(v_1^m)+w(v_2^m)=(e^m)$. These edges enter G'_m, so the tree is no longer Hungarian and can be grown some more.

If, in case 3, condition (h) limits the adjustment at an inner vertex, say v^m, of the tree, it is because, where $\{G_i^a\}$ is the modification of $\{G_i\}$ cited above, $w(v^m)$ becomes as large as the smallest node-weight in B_{mk} of G_{mk}. By substituting $\{G_i^a\}$ for $\{G_i\}$, and calling it $\{G_i\}$, we may regard v^m as u^m, the image of B_{m-1} in G_m of $\{G_i\}$.

Node-weight $w(u^m)$ was first set equal to the smallest node weight $w(q^{m-1})$ in B_{m-1} of G_{m-1}. However, by some weight adjustment where u^m was an outer vertex of a Hungarian tree, $w(u^m)$ may have gotten smaller for a while—before the current adjustment gave $w(u^m)=w(q^{m-1})$ again.

We now describe what to do when u^m, the image of B_{m-1}, is an inner vertex of the current tree and $w(u^m)=w(q^{m-1})$. The pre-image in G_{m-1} of the tree-edges is not generally one tree in G_{m-1} but two. However, as described in section 7 of [2], using only edges from B_{m-1}, this pre-image can always be completed to a planted tree, say T, in G'_{m-1}, where G'_{m-1} is the subgraph of G_{m-1} consisting of edges e^{m-1} for which $w(v_1^{m-1})+w(v_2^{m-1})=w(e^{m-1})$. Thus, we abandon G_m, replace $\{G_i\}$ $(i=0, \ldots, m)$ by $\{G_i\}$ $(i=0, \ldots, m-1)$, and continue the algorithm with respect to T in G'_{m-1}.

This completes the description of the situations and operations in the algorithm. After handling a number of impediments like described in case 3 and like blossoms to be shrunk, the tree will grow into one which allows a decrease in the number of nodes violating (k). Finally when (k) is no longer violated in the final term of $\{G_i\}$, we have a $\{G_i\}$ satisfying all the conditions of theorem (M). It is easy to verify that all the conditions (a) through (j) are preserved by the steps of the algorithm.

The progress of the algorithm is measured first according to the decrease in the number of node weights which violate condition (k). The algorithm consists, in the large, of "growing" trees, one after the other. Each tree is abandoned when and only when it yields a decrease of one or two in that number. Thus, at most N trees are grown where N is the number of nodes in G.

The progress in the growth of an individual tree can be measured according to the number of distinct edges which have entered the tree, including those which have disappeared into blossoms shrunken while growing the tree. The latter never reappear in the same tree because they are absorbed into outer vertices of the tree, whereas only inner vertices of the tree are expanded in the course of its growth. The total number edges which ever enter a tree is less than N, and each edge enters at most once. It can be shown by a survey of the arithmetic involved and the combinatorics involved as described in [2] that an ample upper bound on the order of work between additions of edges to a tree and in the transition from each tree is N^2. This is assuming that the work involved in the individual arithmetic additions and subtractions is fixed. The amount of work in the algorithm also increases some according to the number of significant decimal places in the edge-weights.

Section 8

THEOREM (P) generalizes to the following: *All the extreme points of polyhedra* I *and* II *consist of integer components.* As before the variables $x \epsilon E$ correspond to the edges of a graph G. Now there is an integer "capacity", d, associated with each node $v \epsilon G$. Where all $d=1$, both I and II are polyhedron C of theorem (P).

I. (1) $x \geq 0$, for all $x \epsilon E$.

(2) $\Sigma x \leq d$ $(x \epsilon V)$, for all V, where V corresponds to the edges which meet a node v of G and where d is the integer capacity assigned to v.

(3) $\Sigma x \leq r (x \epsilon R)$, for all R and r, where r is a positive integer, and where R corresponds to the edges of G with both ends in a set S of nodes such that $\Sigma d = 2r+1$ (d summed over capacities of nodes $v \epsilon S$).

II. (1) $0 \leq x \leq 1$, for all $x \epsilon E$.

(2) $\Sigma x \leq d$ $(x \epsilon V)$, for all V, same as in I.

(3) $\Sigma x \leq r$ $(x \epsilon R)$, for all R and r, where r is a posi-

tive integer, and where R corresponds to all edges with both ends in a set S plus a number t of edges with one end in S, where less than d of the t edges meet node v in S, and where $2r+1=t+\Sigma d$ (d summed over capacities of nodes $v \epsilon S$).

Correspondingly, there is an efficient algorithm for finding maximum-weight-sum, degree-constrained subgraphs.

(Paper 69B1–143)

References

[1] G. Birkhoff, Tres. observaciones sobre el algebra lineal, Rev. Univ. Nac. Tucuman, Ser. A, **5,** 147–148 (1946).

[2] J. Edmonds, Paths, trees, and flowers, to appear in the Canadian J. Math. (May 1965).

[3] H. W. Kuhn, Variants of the Hungarian method for assignment problems, Naval Res. Logist. Quart. **3,** 253–258 (1956).

[4] W. T. Tutte, A short proof of the factor theorem for finite graphs, Canadian J. Math. **6,** 347–352 (1954).

[5] J. Von Neuman, A certain zero-sum two-person game equivalent to the optimum assignment problem, in Contributions to the Theory of Games, II (Ann. Math. Studies **38,** 5–12, ed. Tucker and Kuhn), Princeton (1953).

[6] C. Witzgall and C. Zahn, Jr., A modification of Edmonds' maximum matching algorithm, J. Res. NBS 69B (Math. and Math. Phys.) No. 1,