PATHS, TREES, AND FLOWERS

JACK EDMONDS

1. Introduction. A *graph G* for purposes here is a finite set of elements called *vertices* and a finite set of elements called *edges* such that each edge *meets* exactly two vertices, called the *end-points* of the edge. An edge is said to *join* its end-points.

A *matching* in G is a subset of its edges such that no two meet the same vertex. We describe an efficient algorithm for finding in a given graph a matching of maximum cardinality. This problem was posed and partly solved by C. Berge; see Sections 3.7 and 3.8.

Maximum matching is an aspect of a topic, treated in books on graph theory, which has developed during the last 75 years through the work of about a dozen authors. In particular, W. T. Tutte **(8)** characterized graphs which do not contain a *perfect* matching, or *1-factor* as he calls it—that is a set of edges with exactly one member meeting each vertex. His theorem prompted attempts at finding an efficient construction for perfect matchings.

This and our two subsequent papers will be closely related to other work on the topic. Most of the known theorems follow nicely from our treatment, though for the most part they are not treated explicitly. Our treatment is independent and so no background reading is necessary.

Section 2 is a philosophical digression on the meaning of "efficient algorithm." Section 3 discusses ideas of Berge, Norman, and Rabin with a new proof of Berge's theorem. Section 4 presents the bulk of the matching algorithm. Section 7 discusses some refinements of it.

There is an extensive combinatorial-linear theory related on the one hand to matchings in bipartite graphs and on the other hand to linear programming. It is surveyed, from different viewpoints, by Ford and Fulkerson in **(5)** and by A. J. Hoffman in **(6)**. They mention the problem of extending this relationship to non-bipartite graphs. Section 5 does this, or at least begins to do it. There, the König theorem is generalized to a matching-duality theorem for arbitrary graphs. This theorem immediately suggests a polyhedron which in a subsequent paper **(4)** is shown to be the convex hull of the vectors associated with the matchings in a graph.

Maximum matching in non-bipartite graphs is at present unusual among combinatorial extremum problems in that it is very tractable and yet not of the "unimodular" type described in **(5** and **6)**.

Received November 22, 1963. Supported by the O.N.R. Logistics Project at Princeton University and the A.R.O.D. Combinatorial Mathematics Project at N.B.S.

Section 6 presents a certain invariance property of the dual to maximum matching.

In paper **(4)**, the algorithm is extended from maximizing the cardinality of a matching to maximizing for matchings the sum of weights attached to the edges. At another time, the algorithm will be extended from a capacity of one edge at each vertex to a capacity of d_i edges at vertex v_i.

This paper is based on investigations begun with G. B. Dantzig while at the RAND Combinatorial Symposium during the summer of 1961. I am indebted to many people, at the Symposium and at the National Bureau of Standards, who have taken an interest in the matching problem. There has been much animated discussion on possible versions of an algorithm.

2. Digression. An explanation is due on the use of the words "efficient algorithm." First, what I present is a conceptual description of an algorithm and not a particular formalized algorithm or "code."

For practical purposes computational details are vital. However, my purpose is only to show as attractively as I can that there is an efficient algorithm. According to the dictionary, "efficient" means "adequate in operation or performance." This is roughly the meaning I want—in the sense that it is conceivable for maximum matching to have no efficient algorithm. Perhaps a better word is "good."

I am claiming, as a mathematical result, the existence of a *good* algorithm for finding a maximum cardinality matching in a graph.

There is an obvious finite algorithm, but that algorithm increases in difficulty exponentially with the size of the graph. It is by no means obvious whether *or not* there exists an algorithm whose difficulty increases only algebraically with the size of the graph.

The mathematical significance of this paper rests largely on the assumption that the two preceding sentences have mathematical meaning. I am not prepared to set up the machinery necessary to give them formal meaning, nor is the present context appropriate for doing this, but I should like to explain the idea a little further informally. It may be that since one is customarily concerned with existence, convergence, finiteness, and so forth, one is not inclined to take seriously the question of the existence of a *better-than-finite* algorithm.

The relative cost, in time or whatever, of the various applications of a particular algorithm is a fairly clear notion, at least as a natural phenomenon. Presumably, the notion can be formalized. Here "algorithm" is used in the strict sense to mean the idealization of some physical machinery which gives a definite output, consisting of cost plus the desired result, for each member of a specified domain of inputs, the individual problems.

The problem-domain of applicability for an algorithm often suggests for itself possible measures of size for the individual problems—for maximum matching, for example, the number of edges or the number of vertices in the

graph. Once a measure of problem-size is chosen, we can define $F_A(N)$ to be the least upper bound on the cost of applying algorithm A to problems of size N.

When the measure of problem-size is reasonable and when the sizes assume values arbitrarily large, an asymptotic estimate of $F_A(N)$ (let us call it *the order of difficulty of algorithm* A) is theoretically important. It cannot be rigged by making the algorithm artificially difficult for smaller sizes. It is one criterion showing how good the algorithm is—not merely in comparison with other given algorithms for the same class of problems, but also on the whole how good in comparison with itself. There are, of course, other equally valuable criteria. And in practice this one is rough, one reason being that the size of a problem which would every be considered is bounded.

It is plausible to assume that any algorithm is equivalent, both in the problems to which it applies and in the costs of its applications, to a "normal algorithm" which decomposes into elemental steps of certain prescribed types, so that the costs of the steps of all normal algorithms are comparable. That is, we may use something like Church's thesis in logic. Then, it is possible to ask: Does there or does there not exist an algorithm of given order of difficulty for a given class of problems?

One can find many classes of problems, besides maximum matching and its generalizations, which have algorithms of exponential order but seemingly none better. An example known to organic chemists is that of deciding whether two given graphs are isomorphic. For practical purposes the difference between algebraic and exponential order is often more crucial than the difference between finite and non-finite.

It would be unfortunate for any rigid criterion to inhibit the practical development of algorithms which are either not known or known not to conform nicely to the criterion. Many of the best algorithmic ideas known today would suffer by such theoretical pedantry. In fact, an outstanding open question is, essentially: "how good" is a particular algorithm for linear programming, the simplex method? And, on the other hand, many important algorithmic ideas in electrical switching theory are obviously not "good" in our sense.

However, if only to motivate the search for good, practical algorithms, it is important to realize that it is mathematically sensible even to question their existence. For one thing the task can then be described in terms of concrete conjectures.

Fortunately, in the case of maximum matching the results are positive. But possibly this favourable position is very seldom the case. Perhaps the twoness of edges makes the algebraic order for matching rather special in comparison with the order of difficulty for more general combinatorial extremum problems (cf. **3**).

An upper bound on the order of difficulty of the matching algorithm is n^4, where n is the number of vertices in the graph. The algorithm consists of "growing" a number of trees in the graph—at most n—until they augment or

become Hungarian. A tree is grown by branching from a vertex in the tree to an edge-vertex pair not yet in the tree—at most n times. Such a branching may give rise to a back-tracing through at most n edge-vertex pairs in the tree in order to relabel some of them as forming a blossom or an augmenting path. At each of these three levels there may be other labelling work involved—but it is majorized by the work already cited. The work of identifying and labelling the vertex at the other end of some edge to a given vertex need not increase more than linearly with n.

An upper bound on the order of magnitude of memory needed for the algorithm is n^2—the same order of magnitude of memory used to store the graph itself.

3. Alternating paths.

3.0. A *subgraph* of graph G is a graph consisting of a subset of vertices in G and a subset of edges in G under the same incidences which hold for them in G. A non-empty graph G is called *connected* if there is no pair of non-empty subgraphs of G such that each vertex of G and each edge of G is contained in exactly one of the subgraphs. The vertices and edges of any graph partition uniquely into zero or more connected subgraphs, called its *components*.

Maximum, *minimum*, and *odd* will refer to cardinality unless otherwise stated.

3.1. The graph E, *formed* from a set E of edges in G, is the subgraph of G consisting of edges E and their end-points. Any graph H, unless it has a single-vertex component, is formed by its edges. Thus in some contexts it causes no confusion to make no explicit distinction between a graph and its edge-set. In particular, a matching in G may be thought of as a subgraph of G whose components are distinct edges. The sum of two sets D and E is commonly defined as $D + E = (D - E) \cup (E - D)$. The sum $D + E$ of two graphs D and E, formed by edge-sets D and E, is defined to be the graph formed by the edge-set $D + E$.

3.2. There are two other kinds of subtraction for graphs besides the set-theoretic difference used above. With these we must distinguish between a subgraph and the edges which form it. Where G is a graph and E is a set of edges, $G - E$ is the subgraph of G consisting of all the vertices of G and the edges of G not in E. For two graphs G and H, $G - H$ is the subgraph of G consisting of the vertices of G not in H and the edges of G not meeting vertices of H.

Graph $G \cup H$ (graph $G \cap H$) consists of the union (intersection) of the vertex-sets and the edge-sets of graphs G and H, with incidences in $G \cup H$ (graph $G \cap H$) the same as in G and H. We may also take the intersection or union of a graph with a set of edges to get, respectively, a set of edges or a graph. In the latter case the end-points of the edges being adjoined to the

graph must be specified. We shall have occasion to give the same edge different end-points in different graphs.

3.3. A *circuit* B in graph G is a connected subgraph in which each vertex of B meets exactly two edges of B. A (simple) *path* P in G is either a single vertex (joining itself to itself) or else a connected subgraph whose two *end-points* each meet one edge, an *end-edge*, of P and whose other vertices each meet two edges of P. A path is said to *join* its end-points.

3.4. For the pair (G, M), where M is a matching in G, a vertex is called *exposed* if it meets no edge of M. Let $\bar{M}$ denote the edges of G not in M. Define an *alternating* path or *alternating* circuit, P, in (G, M) to be such that one edge in $M \cap P$ and one edge in $\bar{M} \cap P$ meets each vertex of P, except the end-points in the case of a path. Several authors, beginning with J. Peterson in 1891, have used alternating paths to prove the existence of "factors" in certain kinds of graphs.

3.5. *For any two matchings M_1 and M_2 in G, the components of the subgraph formed by $M_1 + M_2$ are paths and circuits which are alternating for (G, M_1) and for (G, M_2). Each path end-point is exposed for either M_1 or M_2.*

A vertex of G meets no more than one edge, each, of M_1 and M_2—and thus no more than two edges of $M_1 + M_2$, one in $M_1 \cap \bar{M}_2$ and one in $M_2 \cap \bar{M}_1$. An end-point v of a path in graph $M_1 + M_2$, meeting an end-edge in $M_1 \cap \bar{M}_2$, say, meets no other edge of M_1. Hence, if an edge of M_2 meets v, it does not belong to M_1 and so it *does* belong to $M_1 + M_2$. But then v is not an end-point. Therefore v is exposed for M_2. This completes the proof.

3.6. An alternating path A in (G, M) joining two exposed vertices contains one more edge of $\bar{M}$ than of M. $M + A$ is a matching of G larger than M by one. Such a path is called *augmenting*. Thus matching M is not maximum if (G, M) contains an augmenting path. The converse also holds:

3.7 (Berge, **1**). *A matching M in G is not of maximum cardinality if and only if (G, M) contains an alternating path joining two exposed vertices of M.*

If M_2 is a larger matching than M, some component of graph $M + M_2$ must contain more M_2-edges than M. By 3.5, such a component is an augmenting path for (G, M).

3.8. Berge proposed searching for augmenting paths as an algorithm for maximum matching. In fact, he proposed to trace out an alternating path from an exposed vertex until it must stop and, then, if it is not augmenting, to back up a little and try again, thereby exhausting possibilities.

His idea is an important improvement over the completely naive algorithm. However, depending on what further directions are given, the task can still be one of exponential order, requiring an equally large memory to know when it is done.

Norman and Rabin **(7)** present a similar method for finding in G a minimum *cover-by-edges*, C, a minimum cardinality set of edges in G which meets every vertex in G. The Berge-Norman-Rabin theorem **(2)** is generalized in **(3)**, but a corresponding generalization of the algorithm presented here in Section 4 is unknown.

3.9. Norman and Rabin also show that the maximum matching problem and the minimum cover-by-edges problem are equivalent.

Assuming every vertex meets an edge, the minimum cardinality of a cover of the vertices in G by a set of edges equals the minimum cardinality of a cover of the vertices in G by a set of edges and vertices, where a vertex is regarded as covering itself. By replacing edges by vertices or vice versa, one can go back or forth between a minimum cover by a set of edges and a minimum cover by a set of edges and vertices, where the latter set consists of a maximum matching together with its exposed vertices.

4. Trees and flowers.

4.0. A *tree* may be defined as (1) a graph T every pair of whose vertices is joined by exactly one path in T; (2) inductively, as either a single vertex or else the union of two disjoint trees together with an edge which has one end-point in each; (3) as a connected graph with one more vertex than edges; and so on.

4.1. An *alternating tree* J is a tree each of whose edges joins an *inner* vertex to an *outer* vertex so that each inner vertex of J meets exactly two edges of J. *An alternating tree contains one more outer vertex than inner vertices.* This follows from the third definition of tree by regarding each inner vertex with its two edges as a single edge joining two outer vertices.

4.2. *For each outer vertex v of an alternating tree J there is a unique maximum matching of J which leaves v exposed and the only exposed vertex in J. Every maximum matching of J is one of these.*

Definition (2) of tree can be strengthened to the statement that a tree minus any one of its edges is two trees. Thus J minus any one of its inner vertices, say u, is two alternating trees. One of these, J_1, contains v as an outer vertex. Assume inductively that J_1 can be matched uniquely so only v is exposed and that J_2, the other subtree, can be matched uniquely so only the vertex v_2, joined in J to u by edge e_2, is exposed. Then the union of e_2 and these two matchings is a matching of J which leaves only v exposed. Since every edge of J has one inner and one outer end-point, every maximum matching leaves only an outer vertex exposed.

4.3. A *planted tree*, $J = J(M)$, of G for matching M is an alternating tree in G such that $M \cap J$ is a maximum matching of J and such that the vertex r

in J which is exposed for $M \cap J$ is also exposed for M. That is, all matching edges which meet J are in J. Vertex r is called the *root* of $J(M)$.

In planted tree $J(M)$ every alternating path $P(M)$, which has outer vertex v and the matching edge to v at one of its ends, is a subpath of the alternating path $P_v(M)$ in $J(M)$ which joins v to the root r.

For $k \geqslant 1$, assume that P_{2k-1} is the unique path $P(M)$ which contains $2k - 1$ edges and assume that at its non-v end it has an inner vertex u_k and a matching edge. Then P_{2k}, consisting of P_{2k-1} together with the unique non-matching edge in J which meets u_k, is the unique path $P(M)$ with $2k$ edges. It has outer vertices v_k and v at its two ends. If $v_k \neq r$, then P_{2k+1}, consisting of P_{2k} together with the unique matching edge which meets v_k, is the unique path $P(M)$ with $2k + 1$ edges. It has an inner vertex u_{k+1} and a matching edge at its non-v end. Since our assumption is true for $k = 1$ and since k cannot become infinite, the theorem follows by induction.

We define a *stem* in (G, M) as either an exposed vertex or an alternating path with an exposed vertex at one end and a matching edge at the other end. The exposed vertex and the vertex at the other end are, respectively, the *root* and the *tip* of the stem. The preceding theorem tells us that (1) no trial-and-error search is required to find the path in J from any of its vertices back to the root and (2) the path P_v in J joining any outer vertex v to the root of J is a stem.

4.4. An *augmenting tree*, $J_A = J_A(M)$, in (G, M) is a planted tree $J(M)$ plus an edge e of G such that one end-point of e is an outer vertex v_1 of J and the other end-point v_2 is exposed and not in J. *The path in J_A which joins v_2 to the root of J is an augmenting path.* This follows immediately from (4.3).

4.5. For each vertex b of an odd circuit B there is a unique maximum matching of B which leaves b exposed. A *blossom*, $B = B(M)$, in (G, M) is an odd circuit in G for which $M \cap B$ is a maximum matching in B with say vertex b exposed for $M \cap B$. A *flower*, $F = F(M)$, consists of a blossom and a stem which intersect only at the tip of the stem (the vertex b).

A *flowered tree*, J_F, in (G, M) is a planted tree J plus an edge e of G which joins a pair of outer vertices of J. *The union of e and the two paths which join its outer-vertex end-points to the root of J is a flower, F.*

Let v_1 and v_2 be these outer vertices, and P_1 and P_2 be the paths in J joining them to r. We have seen that P_1 and P_2 are stems (which are easily recovered from J). Since they intersect in at least r and since the path in J joining r to any other vertex is unique, $P_b = P_1 \cap P_2$ is an alternating path with an end at r. If its other end-point, say b, were inner, it would be distinct from r, v_1, and v_2. Thus r would be distinct from v_1 and v_2, and b would meet three different edges of J, one in P_b, one in P_1 not in P_b, and one in P_2 not in P_b. But an inner vertex meets only two edges in the tree. Therefore b is outer and P_b is a stem. Thus $P_1' = P_1 - (P_b - b)$ and $P_2' = P_2 - (P_b - b)$, unless one is a

vertex $v_1 = b$ or $v_2 = b$, have non-matching edges at their b-ends and matching edges at their outer ends. It follows in any case that $B = P_1' \cup P_2' \cup e$ is a circuit with only b exposed for $M \cap B$, and thus B is a blossom with p_b as its stem.

4.6. A *Hungarian tree* H in a graph G is an alternating tree whose outer vertices are joined by edges of G only to its inner vertices.

4.7. *For a matching M in a graph G, an exposed vertex is a planted tree. Any planted tree $J(M)$ in G can be extended either to an augmenting tree, or to a flowered tree, or to a Hungarian tree* (merely by looking at most once at each of the edges in G which join vertices of the final tree).

An exposed vertex satisfies the definition of planted tree. Suppose we are given a planted tree J and a set D (perhaps empty) of edges in G which are not in J but which join outer to inner vertices of J. (1) If no outer vertex of J meets an edge not in $D \cup J$, then J is Hungarian. Suppose outer vertex v_1 meets an edge e not in $D \cup J$, whose other end-point is, say, v_2. (2) If v_2 is an inner vertex of J, we can enlarge D by adjoining e. (3) If v_2 is an outer vertex of J, then $e \cup J$ is a flowered tree. (4) If v_2 is exposed and not in J, then $e \cup J$ is an augmenting tree. (5) Finally, if v_2 is not exposed and not in J, then the M-edge e_2 which meets v_2 is not in J, and thus v_3, the other end-point of e_2, is not in J by the definition of planted tree. Therefore, in this case we can extend J to a larger planted tree with new inner vertex v_2 and new outer vertex v_3 by adjoining edges e and e_2. For any J and D, one of the five cases holds. Therefore by looking at any edge in G at most once, we can reach one of the three cases described in the theorem, because the other two cases, (2) and (5), consume edges and G is finite.

4.8. The algorithm which is being constructed is efficient because it does not require tracing many various combinations of the same edges in order to find an augmenting path or to determine that there are none. In fact we accomplish one or the other without ever looking again at the edges encountered in process (4.7), except to pick out from the tree the blossom or the augmenting path when case (3) or (4) occurs. We see from (4.3) and (4.5) how easy it is to retrieve the blossom or the path. When flowers arise we "shrink" the blossoms, and so if an augmenting path arises later, it will be in a "reduced" graph. However, only one other very simple kind of task translates the augmentation to (G, M) itself. That task is to expand a shrunken blossom to an odd circuit and find the maximum matching of the odd circuit which leaves a certain vertex exposed. Actually, we shall find in (7.3) that it is desirable to leave odd circuits shrunk while looking in the reduced graph for as many successive augmentations as possible since they are all reflected in augmentations of (G, M).

4.9. For H, a subgraph of G, G is the disjoint union $(G - H) \cup \delta H \cup H^+$, where δH is the set of the edges with one end-point in H and one end-point in

$G - H$, and where $H^+ = G - (G - H)$ is the subgraph consisting of H and all edges of G with both end-points in H. When H is connected, *shrinking* H means constructing the new graph $G/H = (G - H) \cup \delta H \cup h$ by regarding H^+ as a single new vertex, $h = H/H$, which meets the edges $\delta H = \delta h$. The end-points in $G - H$ of the edges δH do not change.

4.10. If B is an odd circuit in G, then $b' = B/B$ is called a *pseudovertex* of G/B. To *expand* b' means to recover G from G/B. The algorithm, after it expands a pseudovertex b', will make use of the circuit B. In general, finding a "Hamiltonian" circuit in a graph B^+ is difficult. Therefore, when the algorithm shrinks B to form G/B, it should remember circuit B as having effected the shrinking. Thus we call circuit B in G (rather than B^+) the *expansion of* b'. In formal calculation shrinking B in G is an easy operation. Essentially, just assign all the vertices and edges of B a label, b', and then, until b' is expanded by erasing these labels, ignore any distinction between vertices labelled b' and ignore edges joining them to each other.

Where M is a matching set of edges in G, M/B is defined as $M \cap (G/B)$. Clearly, if B is a blossom for (G, M), then M/B is a matching of G/B.

4.11. Let $G_0 = G$, $G_i = G_{i-1}/B_i$, and $b_i = B_i/B_i$ for $i = 1, \ldots, n$, where B_i is an odd circuit in graph G_{i-1}. We inductively define the *pseudovertices* (with respect to G) of G_k ($k = 1, \ldots, n$) to be b_k together with the pseudovertices in $G_{k-1} - B_k = G_k - b_k$. Of course not every b_i, $i \leqslant k$, will be a pseudovertex of G_k because some will have been absorbed into others. The order in which the pseudovertices of a G_k arise is immaterial. That is, the order in which the odd circuits B_i are shrunk is immaterial except in so far as one shrunken B_i is a vertex in another B_i. Thus we can expand any pseudovertex b_j of G_k to obtain a graph G_{kj} for which $G_{kj}/B_j = G_k$. The pseudovertices of G_{kj} (with respect to G) are the pseudovertices in B_j together with the pseudovertices in $G_k - b_j$; that is, graph G_{kj} can be obtained from G by shrinking in a proper order the odd circuits which were absorbed into these pseudovertices. On the other hand, we do not expand a vertex b_h in B_k until vertex b_k is expanded.

There is a partial order on the b_i's defined by the transitive completion of the relation $b_h < b_k$ where b_h is a vertex of B_k. (It is a special kind of partial ordering because each b_h is a vertex of at most one B_k.) There is a partial order on the sets, $S_\alpha, S_\beta, \ldots$, of mutually incomparable b_i's, where $S_\alpha \leqslant S_\beta$ when every member of S_α is less than or equal to some member of S_β. Evidently there is a unique family of graphs, $G_\alpha, G_\beta, \ldots$, which include the G_i's and G. They correspond 1–1 to the sets, $S_\alpha, S_\beta, \ldots$, so that the pseudovertices of G_α are S_α, etc. We have $S_\alpha \leqslant S_\beta$ if and only if G_β can be obtained from G_α by shrinking certain B_i, those for which b_i is less than or equal to some member of S_β and not less than or equal to any member of S_α. Graph G corresponds to the empty set and G_n corresponds to the set of b_i's which are maximal with respect to their partial order.

The *complete expansion* of a pseudovertex b_i is the subgraph

$$U^+ = G - (G - U) \subset G$$

where U consists of all vertices of G absorbed into b_i by shrinking.

4.12. *Where B is the blossom of a flower F for (G, M), M is a maximum matching of G if and only if M/B is a maximum matching of G/B.*

4.13. *Where blossom B is in J_F, a planted flowered tree for (G, M), J_F/B is a planted tree for $(G/B, M/B)$. It contains B/B as an outer vertex. Its other outer and inner vertices are respectively those of J_F which are not in B.*

Theorem (4.13) follows easily from (4.5). We separate the two converse statements of Theorem (4.12) into slightly stronger statements, (4.14) and (4.15).

4.14. *Where B is any odd circuit in G, for every matching M_1 of G/B there exists a maximum matching M_B of B such that $M = M_1 \cup M_B$ is a matching for G.*

Since any matching M_1 of G/B contains at most one edge meeting B/B, the edges M_1 in G meet at most one vertex, say b_1, of B. Therefore the desired M_B is the maximum matching of B which leaves b_1 exposed. Since the cardinality $|M_B|$ of M_B is constant, any augmentation of M_1 yields a corresponding augmentation of M. Therefore, the "only if" part of (4.12) is proved.

Applying the above matching operation to successive expansions of pseudovertices into odd circuits we have:

Where P is the complete expansion of a pseudovertex p in G_2, where G_1 is the graph obtained from G_2 by completely expanding p, and where M_2 is any matching of G_2, there exists a matching M_P of P leaving exactly one exposed vertex in P such that $M_P \cup M_2$ is a matching of G_1. Thus since $|M_P|$ is constant, any augmentation in G_2 yields a corresponding augmentation in G_1.

4.15. *For (G, M), let P be a subgraph such that* (1) *$M \cap P$ leaves exactly one exposed vertex in P,* (2) *M/P is a maximum matching of G/P, and* (3) *$p = P/P$ is the tip of a stem S_p for $(G/P, M/P)$. Then M is a maximum matching of G.*

The edges of S_p form in G a stem, S, for (G, M). (In case S_p has no edges, take S to be the vertex in P exposed for M.) Compare $M' = M + S$ and M'/P with M and M/P. The definition of stem implies that M' is a matching of G with $|M'| = |M|$ and that the exposure of the root of S is changed to the exposure of the tip of S. Similarly $M'/P = M/P + S_p$ is a matching of G/P with $|M'/P| = |M/P|$ and with vertex p exposed. Because the cardinalities do not change, it is sufficient to show that M' is maximum in G if M'/P is maximum in G/P.

Using (3.7), if M' is not maximum, G contains an augmenting path $A = A(M')$. If A contains no vertices of P, then it is also an augmenting

path for M'/P in G/P. Otherwise, because P contains only one exposed vertex for M', at least one of the ends of A is at an exposed vertex u_1 not in P. There is a unique subpath A_1 of A with one end-point at u_1 and containing only one vertex p_1 of P, at its other end. The only difference between A_1 and $A_1/P = (A_1 \cup P)/P$ is that p_1 is replaced by p, which is exposed for M'/P. Thus A_1/P is an augmenting path for M'/P and so M'/P is not maximum. The theorem is proved.

The theorem extends as follows:

For (G, M), let $P_1, \ldots, P_n$ be a family of disjoint subgraphs in G such that (1) $M \cap P_i$ leaves exactly one exposed vertex in P_i, (2) $M_n = M \cap G_n$ is a maximum matching of $G_n = G/P_1/\ldots/P_n$, and (3) vertices P_i/P_i of G_n are outer vertices in a planted tree J_n for (G_n, M_n). Then M is a maximum matching of G.

We may assume that the indices order the P_i/P_i's so that (for $k = 1, \ldots, n-1$) those from 1 through k are contained in a planted subtree J_k of J_n not containing those from $k+1$ through n. Hence the theorem follows by induction after proving that $M_{n-1} = M \cap G_{n-1}$ is a maximum matching of $G_{n-1} = G/P_1/\ldots/P_{n-1}$. Since every outer vertex of J_n is the tip of a stem in G_n, this follows from the last theorem.

4.16. Theorems (4.7) and (4.13) show how by branching a planted tree out from an exposed vertex of (G, M) and shrinking blossoms B_i when they are encountered, we eventually obtain in a graph $G_k = G/B_1/\ldots/B_k$ either a tree with an augmenting path or a Hungarian tree. An augmenting path admits an augmentation of matching $M_k = M \cap G_k$ according to (3.7), and (4.14) shows how this induces an augmentation of matching $M_{k-1} = M \cap G_{k-1}$ and so on back through M. On the other hand, when a Hungarian tree J is obtained, submatching $(J \cup B_k \cup \ldots \cup B_1) \cap M$ of (G, M) cannot be improved and so this part of G is freed from further consideration. This follows immediately from (4.15) and the next theorem, (4.17), where G_k is denoted simply as G.

4.17. *Let J be a Hungarian tree in a graph G. A matching M_1 of $G - J$ is maximum in $G - J$ if and only if M_1 together with any maximum matching M_J of J is a maximum matching of G.*

Since J and $G - J$ are disjoint, if there exists a matching M_1' of $G - J$ which is larger than M_1, then $M_1' \cup M_J$ is a larger matching of G than $M_1 \cup M_J$. Conversely, suppose M_1 is maximum for $G - J$. Let

$$M' = M_1' \cup M_I \cup M_J'$$

be an arbitrary matching of G where $M_1' \subset G - J$, where $M_J' \subset J$, and where $M_I \cap ((G - J) \cup J)$ is empty. Then $|M_1'| \leqslant |M_1|$. Every edge in M_I meets at least one inner vertex of J; that is, where $I' \subset I(J)$ is the set of the inner vertices met by M_I, $|M_I| \leqslant |I'|$. The graph $J - I'$ consists of $|I'| + 1$

disjoint alternating trees whose inner vertices together are $I(J) - I'$. Therefore, since the maximum matching cardinality of an alternating tree equals the number of its inner vertices, $|M_J'| \leqslant |I(J) - I'|$. Adding the three inequalities gives $|M'| \leqslant |M_1| + |I(J)| = |M_1 \cup M_J|$. So the theorem is proved.

4.18. The matching M of $G = G^0$, to begin with, may be empty. If it leaves any exposed vertices, then the process (4.16) operates with respect to one of them. Either it produces an augmentation of M by one edge, thus disposing of two exposed vertices, or it reduces the possible domain for augmenting M to a subgraph $G^1 = G_k - J$ of G, containing one less exposed vertex and containing only edges and vertices not previously considered. Successive application of (4.16) may reduce the consideration of M to a subgraph G^i of G and reveal there an augmentation of M. After augmenting in G^i, obtaining a larger M for G with two less exposed vertices in G^i, (4.16) operates again in G^i, never returning to the matching in the rest of G.

4.19. Repeated application of (4.18) reduces the domain in question to a G^n containing no exposed vertices. Then we know that we have a maximum matching; let us still call it M, with n exposed vertices in G. Thus the construction of an algorithm for finding a maximum cardinality matching in a graph is complete. Often the last application of (4.18) is unnecessary. For verifying maximality, the algorithm may as well stop when it reduces the domain to a G^{n-1} containing one exposed vertex, since two exposed vertices are necessary in order to augment. However, for theoretical purposes it is convenient to have the algorithm grow a tree from each exposed vertex of the final, maximum matching.

4.20. We may define an *alternating forest* to be a family of disjoint alternating trees and a *planted forest* in (G, M) to be a family of disjoint planted trees in (G, M). A *dense* planted forest is one which contains all the exposed vertices of (G, M). The family of exposed vertices, itself, is a dense planted forest. The algorithm works as well by growing a dense planted forest all at once, rather than one tree at a time.

It is appropriate then to define *augmenting forest* (*flowered forest*) to be a planted forest plus an edge e of G whose end-points are outer vertices of different trees (of the same tree) of the planted forest.

A *Hungarian forest* in G is defined similarly to Hungarian tree, replacing the word "tree" by "forest." Notice that the trees of a Hungarian forest are not necessarily Hungarian trees—an outer vertex of one tree may be joined by an edge of G to an inner vertex of another tree in the forest.

The theorems on trees presented in this section are essentially the same for forests.

5. The dual to matching.

5.0. A *bipartite* graph K is one in which every circuit contains an even

number of edges. This condition, that K contains no odd circuits, is equivalent to being able to partition the vertices of K into two parts so that each edge of K meets exactly one vertex in each part. The well-known König theorem states:

For a bipartite graph K, the maximum cardinality of a matching in K equals the minimum number of vertices which together meet all the edges of K.

5.1. The linear programming duality theorem states: *If*
(1) $x \geqslant 0$, $Ax \leqslant c$ *and*
(2) $y \geqslant 0$, $A^T y \geqslant b$,
for given real vectors b and c and real matrix A, then for real vectors x and y,

$$\max_x(b, x) = \min_y(c, y)$$

when such extrema exist.

The problems of finding a maximizing vector x and a minimizing vector y are called *linear programmes*, *dual* to each other.

5.2. The König theorem is now widely recognized as the instance of (5.1) where b and c consist of all ones and $A = A_K$ is the zero-one incidence matrix of edges (columns) versus vertices (rows) in a bipartite graph K. In view of Theorem (5.1) the König theorem is equivalent to the remarkable fact that, with b, c, and A as just described, the two linear programmes of (5.1) have solutions x and y whose components are zeros and ones whether or not this condition is imposed. An elegant theory centres on this phenomenon.

Graph-theoretic algorithms are well known for so-called assignment, transportation, and network flow problems **(5)**. These are linear programmes which have constraint matrices A that are essentially A_K.

5.3. For a linear programme with an arbitrary matrix A of integers, or even of zeros and ones, we cannot say that the extreme values will be assumed, as when $A = A_K$, by vectors with integer components. Therefore, in general when we impose the condition of integrality on x, the equality of the two extrema no longer holds.

In particular, when the maximum matching problem is extended from bipartite to general graphs G, a genuine integrality difficulty is introduced. Our matching algorithm met it by the device of shrinking blossoms.

5.4. The matching algorithm yields a generalization of the König theorem to maximum matchings in G. The new matching duality theorem, in the form "maximum cardinality of a matching in G equals minimum of something else," is also an instance of linear programming duality.

It is reasonable to hope for a theorem of this kind because any problem which involves maximizing a linear form by one of a discrete set of non-negative vectors has associated with it a dual problem in the following sense. The discrete

set of vectors has a convex hull which is the intersection of a discrete set of half-spaces. The value of the linear form is as large for some vector of the discrete set as it is for any other vector in the convex hull. Therefore, the discrete problem is equivalent to an ordinary linear programme whose constraints, together with non-negativity, are given by the half-spaces. The dual (more precisely, a dual) of the discrete problem is the dual of this ordinary linear programme.

For a class of discrete problems, formulated in a natural way, one may hope then that equivalent linear constraints are pleasant even though they are not explicit in the discrete formulation.

5.5. Arising from the definition of a matching—no more than one matching edge to each vertex—are the obvious linear constraints that for each vertex $v \in G$ the sum of the x's corresponding to edges which meet v is less than one. To obtain a maximum cardinality matching, we want to maximize the sum of all the x's, corresponding to edges of G, subject to the additional condition that each x is zero or one.

It turns out that maximum matching can be turned into linear programming by substituting for the zero-one condition the additional constraints that the x's are non-negative and that for any set R of $2k + 1$ vertices in $G(k = 1, 2, \ldots)$ the sum of the x's which correspond to edges with both end-points in R is no greater than k. The former condition on the x's obviously implies the latter since for no matching in G do more than k matching edges have both ends in R.

The converse—that subject only to the linear constraints, $\sum x_i$ can be maximized by zeros and ones—is not so obvious, but in view of (5.1) it follows from (5.6), the generalized König theorem.

Actually the stronger converse holds—that subject only to these same linear constraints, $\sum c_i x_i$, for any real numbers c_i, can be maximized by zeros and ones. In other words, the polyhedron described by the constraints is, indeed, the convex hull of the zero-one vectors which correspond to matchings in G. We shall not prove this until we take up maximum weight-sum matching in paper **(4)**. Although the convex-hull notion suggested trying to generalize the König theorem, and although the generalization found does suggest the true convex hull, the success of the first suggestion does not necessarily validate the second.

5.6. A set consisting of one vertex in G is said to *cover* an edge e in G if e meets the vertex. The *capacity* of this set is one. A set consisting of $2k + 1$ vertices in $G(k = 1, 2, \ldots)$ is said to *cover* an edge e in G if both end-points of e are in the set. The *capacity* of this set is k. An *odd-set cover* of a graph G is a family of odd sets of vertices such that each edge in G is covered by a member of the family.

MATCHING-DUALITY THEOREM. *The maximum cardinality of a matching in G equals the minimum capacity-sum of an odd-set cover in G.*

It is obvious that the capacity-sum of any odd-set cover in G is at least as large as the cardinality of any matching in G, so we have only to prove the existence in G of an odd-set cover and a matching for which the numbers are equal.

5.7. The theorem holds for a graph which has a perfect matching M—that is, with no exposed vertices—since the odd-set cover consisting of two sets, one set containing one of the vertices and the other set containing all the other vertices, has capacity-sum equal to $|M|$. It also holds for a graph which has a matching with one exposed vertex. Here the odd-set cover may be taken as consisting of one member, the set containing all vertices of the graph. For the case of one exposed vertex, an odd-set cover may also be constructed as in (5.8) by applying the algorithm to construct a Hungarian tree even though it obviously will not result in augmentation.

5.8. Applying the algorithm to (G, M), where $|M|$ is maximum, using some exposed vertex as root, we obtain a graph G' containing a maximally matched Hungarian tree J, a number of whose outer vertices are pseudo. Let S_J consist of all odd sets of the following two types: sets each consisting of one inner vertex in J, and sets each consisting of the vertices in the complete expansion of one pseudovertex of J.

The number of edges of M which a member of S_J covers is *equal* to the capacity of the member. Every edge of M not in $G' - J$ is covered by *exactly* one member of S_J. An edge of G is covered by a member of S_J if and only if it is not in $G' - J$.

Matching $M \cap (G' - J)$ is a maximum matching of $G' - J$ with one less exposed vertex than (G, M). Assuming that $|M \cap (G' - J)|$ equals the capacity of an odd-set cover, say S'_J, of $G' - J$, we have that $|M|$ equals the capacity of $S_J \cup S'_J$, an odd set cover of G. Theorem (5.6) follows by induction on the number of exposed vertices.

5.9. It is evident from the proof that we may require the minimum odd-set cover to have certain other structure—in particular, that each member with more than one vertex contain the vertices of at least one odd circuit in G. With the latter restriction the theorem becomes a strict generalization of the König theorem.

6. Invariance of the dual.

6.0. For any particular application of the algorithm (4) to G, yielding, say, the maximum matching M, we may skip the augmentation steps in (4.16) by regarding the augmented matching as being the one already at hand. This gives a particular application of (4) to G starting with maximum matching M. In the application of the algorithm to (G, M), we can regard all the branchings and blossom shrinkings as taking place without subtracting the trees J_i as they arise. Thus we obtain from (G, M) a graph G^* with a number of pseudo-

vertices which are outer vertices in a sequence $\{J_i\}$ $(i = 1, \ldots, n)$ of disjoint planted trees in G^*, one corresponding to each exposed vertex of (G, M). By expanding all the pseudovertices of G^* completely, we recover the graph G.

6.1. The tree J_i is Hungarian in $G^* - J_1 \ldots - J_{i-1}$, but usually not Hungarian in G^* because an outer vertex of J_i might be joined to an inner vertex of any other tree with a lower index. Hence the partition of the outer and inner vertices into trees J_i depends on the order of their construction. Also non-matching edges which can occur in each tree are not unique. In general, joining outer to inner vertices of a J_i are many other $\bar{M}$ edges which would do as well. The particular blossoms which led to the pseudovertices are also fairly arbitrary. And, finally, the maximum matching is far from unique. However, (6.2) will show that the graph G^* is uniquely determined by G alone.

6.2. *For a* (G, M) *where* M *is any maximum matching, let* G^* *and* $\{J_i\}$ *be obtained from* (G, M) *by* (6.0).

(*a*) *The non-pseudo outer vertices of the* J_i's *and the vertices of the pseudovertex complete expansions, all called the outer vertices* $O(G)$ *of* G, *are precisely the vertices of* G *which are left exposed by some maximum matching of* G.

(*b*) *The inner vertices of the* J_i's, *called the inner vertices* $I(G)$ *of* G, *are precisely those vertices of* G *not in* $O(G)$ *but joined to vertices in* $O(G)$.

(*c*) G^* *is obtained from* G *by shrinking the connected components of* $O(G)^+$, *the subgraph of* G *consisting of vertices* $O(G)$ *and all edges of* G *joining them.*

6.3. We have defined vertex families $O(G)$ and $I(G)$ in terms of particular J_i. The theorem yields definitions dependent only on G itself.

Clearly $O(G^*)$ and $I(G^*)$, defined in terms of the J_i in G^*, are respectively the outer and the inner vertices of the J_i. Notice that the early definitions of inner and outer, for vertices in an alternating tree, are consistent with the definitions for a general graph.

6.4. *Proof of* (6.2), (b) *and* (c). Let the vertex v^* of G^* be joined in G^* to some outer vertex u^* of J_i. Then v^* is a vertex in some $J_h (h \leqslant i)$, since J_i is Hungarian in $G^* - J_1 - \ldots - J_{i-1}$. But v^* cannot be an outer vertex of J_h since u^* is not inner and since J_h is Hungarian in $G^* - J_1 - \ldots - J_{h-1}$. Therefore v^* is inner. It follows that each outer vertex u of G is joined only to inner vertices and to other vertices in the complete expansion of its image u^*. By construction, each inner vertex is joined to an outer vertex of G. Hence, (b) is true.

Since by construction the complete expansion of each outer vertex of G^* is connected, it also follows that the connected components of $O(G)^+$ correspond precisely to outer vertices of G^*. Hence, (c) is true.

6.5. An outer vertex u of G, by definition, either is identical with or is contained in the complete expansion of some outer vertex u^* of, say, J_i. For any maximum matching M_i of alternating tree J_i, $M_i \cup [M \cap (G^* - J_i)]$ is

a maximum matching of G^*, which by (4.14) induces a maximum matching M' of G. Let M_i be the one which leaves u^* exposed. If u^* is pseudo, then by (4.14) M' can be chosen so that u is exposed in the expansion. This proves half of (6.2), (a).

6.6. C. Witzgall suggested the following simplified proof of the converse, viz. that only the outer vertices are ever exposed for a maximum matching. A non-outer vertex v meets an edge e of M. Deleting v and its adjoining edges, $\cup J_i - v$ is a Hungarian forest in $G^* - v$.

If v is inner, then the forest is dense in $G^* - v$. Otherwise it is dense in $G^* - v$ except for one exposed vertex, the other end of e. In either case it follows that $M - e$ is a maximum matching of $G - v$.

Assume that M' is a maximum matching of G which leaves v exposed. Then M' is also a matching of $G - v$. Since M' is larger than $M - e$, we have a contradiction. This completes the proof of (6.2).

6.7. The definition of odd-set cover may be expanded (more than necessary for Theorem (5.6)) to include the possibility of members which are even sets of vertices in G. A set of $2k$-vertices has capacity k and covers the edges which have both end-points in the set. Then, clearly, Theorem (5.6) still holds for this kind of cover.

With this definition of cover, it follows from the uniqueness of G^* that there is a unique *preferred* minimum cover, S^*, for any graph G. The one-vertex members of S^* are the inner vertices of G^*, the other odd members of S^* correspond to the pseudovertices of G^*, and the one even member of S^* consists of the non-inner, non-outer vertices of G^*.

7. Refinement of the algorithm.

7.0. Several possibilities for refining the algorithm suggest themselves.

We could remember an old tree, uprooted by an augmentation, so that when a new rooted tree takes on a vertex in it, we can immediately adjoin a piece of it to the new tree. This appears not worth doing. A tree is easy to grow, easier than selecting from an old tree the piece which may be grafted.

7.1. A quite useful refinement is to leave the pseudovertices of the old tree shrunk until their expansion is necessary. We see from (4.14) that any further augmentation of a matching M' in a graph G' with pseudovertices yields a further augmentation in G just as easily as the first. On the other hand, a maximum matching in G', reached after one or more augmentations, does not necessarily yield a maximum matching of G. The sufficiency part of (4.12) depends on the blossom being part of a flower, whereas the first augmentation in G' uproots the stem.

7.2. However, we may easily observe the circumstance arising in the application of the algorithm to (G', M') where the shrinkage might hide a

possible augmentation in G. It is where a pseudovertex, say b', becomes an inner vertex of the planted tree, say $J' = J'(M')$.

In this case, we obtain a graph G'' from G' by expanding b' to an odd circuit B. The edges of J' form in G'' a subgraph which we still call J'. The set M' is also a matching in G''. One edge of $J' \cap M'$ has an end-point, say b_1, in B. One edge of $J' \cap \bar{M}'$ has an end-point, say b_2, in B. The maximum matching M_B of B which is compatible with M' in G'' leaves b_1 exposed. The vertices b_1 and b_2 partition B into two paths, P_2 even and P_1 odd, which join b_1 and b_2. The graph $J'' = P_2 \cup J'$ is a planted tree in G'' for the matching $M_B \cup M'$.

Unless b_1 and b_2 coincide, P_2 will contain outer vertices of J''. These may be joined to vertices not in J'' which admit an extension of J'', not possible for $J''/B = J' \subset G'$, to a planted tree with an augmenting path.

7.3. If $J' \subset G'$ can be extended in G' to a tree with an augmenting path, it does not matter that some of the inner vertices are pseudo because a further augmentation for G is thus determined. If J' with pseudo inner vertex b' can be extended in (G', M') to a flowered tree whose blossom B' contains b', then b' loses its distinction as an inner vertex. It might as well stay shrunk and be absorbed into the new pseudovertex B'/B' of G'/B'. In fact, Theorems (4.15) and (4.17), together, tell us that any pseudo outer vertex might as well be left pseudo during the algorithm.

Therefore a pseudo inner vertex should be retained until a planted Hungarian tree J_H is obtained. If no inner vertices of J_H are pseudo, then (4.17) is applicable. Otherwise, at this point, a pseudo inner vertex should be expanded according to (7.2).

7.4. One of the main operations of the algorithm is described in (4.3). That is back-tracing along paths in a tree already constructed, either to obtain an augmentation as in (4.4) or to delineate a new blossom as in (4.5). The back-tracing takes place in an alternating tree only because blossoms have been shrunk to pseudovertices. A pseudovertex may be compounded from many earlier blossom shrinkings and may thus encompass a complicated subgraph of G. After shrinking, back-tracing entirely bypasses the internal structure of a pseudovertex.

A possible alternative to actually shrinking is some method for tracing through the internal structure of a pseudovertex. Witzgall and Zahn **(9)** have designed a variation of the algorithm which does that. Their result is attractive and deceptively non-trivial.

References

1. C. Berge, *Two theorems in graph theory*, Proc. Natl. Acad. Sci. U.S., *43* (1957), 842–4.

2. ——— *The theory of graphs and its applications* (London, 1962).

3. J. Edmonds, *Covers and packings in a family of sets*, Bull. Amer. Math. Soc., *68* (1962), 494–9.

4. ——— *Maximum matching and a polyhedron with* (0, 1) *vertices*, appearing in J. Res. Natl. Bureau Standards *69*B (1965).
5. L. R. Ford, Jr. and D. R. Fulkerson, *Flows in networks* (Princeton, 1962).
6. A. J. Hoffman, *Some recent applications of the theory of linear inequalities to extremal combinatorial analysis*, Proc. Symp. on Appl. Math., *10* (1960), 113–27.
7. R. Z. Norman and M. O. Rabin, *An algorithm for a minimum cover of a graph*, Proc. Amer. Math. Soc., *10* (1959), 315–19.
8. W. T. Tutte, *The factorization of linear graphs*, J. London Math. Soc., *22* (1947), 107–11.
9. C. Witzgall and C. T. Zahn, Jr., *Modification of Edmonds' algorithm for maximum matching of graphs*, appearing in J. Res. Natl. Bureau Standards *69*B (1965).

National Bureau of Standards and
Princeton University