MACHINE SEQUENCING VIA DISJUNCTIVE GRAPHS: AN IMPLICIT ENUMERATION ALGORITHM

Egon Balas

Carnegie-Mellon University, Pittsburgh, Pennsylvania

(Received March 11, 1968)

One formulation of the machine sequencing problem is that of finding a minimaximal path in a disjunctive graph. This paper describes an implicit enumeration procedure that solves the problem by generating a sequence of circuit-free graphs and solving a slightly amended critical-path problem for each graph in the sequence. Each new term of the sequence is generated from an earlier one by complementing one disjunctive arc. The search tree is drastically cut down by the fact that the only disjunctive arcs that have to be considered for being complemented are those on a critical path. An evaluation of these candidates is used to direct the search at each stage. The procedure can start with any feasible schedule (like the one actually used in production, or generated by some heuristics), and gradually improve it. Thus one can possibly stop short of the optimum, with a reasonably 'good' feasible schedule. Storage requirements are limited to the data pertinent to the current node of the search tree.

THE MACHINE-sequencing problem (also known as job-shop scheduling, *see* references 1–9), in its simplest form, is that of finding an optimal sequence for processing m items on q machines, when the completion time for each operation is known, and (a) a given operation (job) on a given item has to be performed on a specified machine, (b) the operations pertaining to a given item are to be carried out in a technologically prescribed sequence, (c) there is freedom of choice as to the sequence of operations for each machine, and (d) one is looking for a sequence minimizing total completion time (the time needed to perform all operations on all items).

It is known[8, 10, 11] that this problem reduces to that of finding a minimaximal path in a disjunctive graph.

Two arcs of a graph are said to form a *disjunctive pair*,[8] if any path in the graph is allowed to meet at most one of them. A graph D containing disjunctive arcs is called a *disjunctive graph* and denoted

$$D=(N; Z, W), \tag{1}$$

where N is the set of nodes, Z the set of conjunctive (i.e., nondisjunctive) arcs, and W the set of disjunctive arcs.

For each machine sequencing problem one can define a (directed) disjunctive graph $D=(N; Z, W)$, by associating

(α) a node $j \epsilon N$ with each operation, including two dummies: node 0 ('start') to be the source of D, and node n ('end') to be the sink of D;

(β) a (conjunctive) arc $(i, j) \epsilon Z$ with each pair of operations pertaining to the same item and adjacent in the technological sequence; also, an arc $(0, h) \epsilon Z$ for each h that is the first operation to be performed on some item, and an arc (k, n) for each k that is the last operation pertaining to an item;

(γ) a disjunctive pair of arcs $(i, j) \epsilon W$, $(j, i) \epsilon W$, with each pair of operations to be performed on different items but on the same machine;

(δ) a length (nonnegative real) d_{ij} with each arc $(i, j) \epsilon Z \cup W$, equal to the minimal required time lapse between the starting of operations i and j.

A disjunctive pair of arcs $[(i, j), (j, i)]$ expresses the condition that one of the two operations i, j, must be finished before the other one is started.

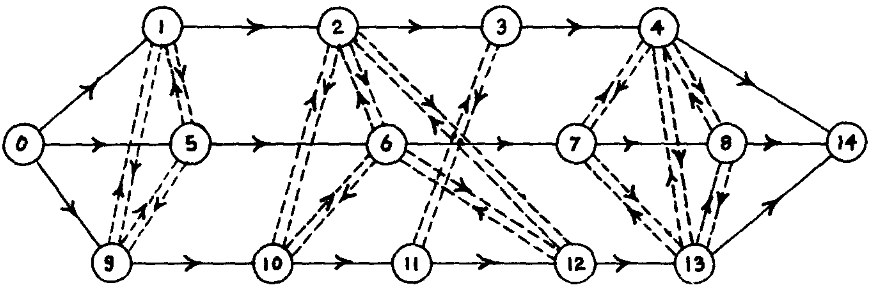

Figure 1

Figure 1 depicts the disjunctive graph of a problem with 3 items and 4 machines. If N_i denotes the operations pertaining to item i, and N^k those pertaining to machine k, then

$$N_1 = \{1, 2, 3, 4\}, \quad N_2 = \{5, 6, 7, 8\}, \quad N_3 = \{9, 10, 11, 12, 13\};$$
$$N^1 = \{1, 5, 9\}, \quad N^2 = \{2, 6, 10, 12\}, \quad N^3 = \{3, 11\}, \quad N^4 = \{4, 7, 8, 13\}.$$

A *path* in D or in any of the graphs considered in this paper will be defined as a sequence of arcs $(i_1, j_1), (i_2, j_2), \cdots, (i_n, j_n)$, such that $j_k = i_{k+1}$ for $k = 1, \cdots, n-1$. A *circuit* is a closed path, i.e., such that $j_n = i_1$. The *length* of a path is the sum of the lengths of the arcs in the path. If (i, j) is an arc of the graph, i is a *predecessor* of j, and j is a *successor* of i.

Denoting by $G = (N, Z)$ the graph obtained from D by dropping all disjunctive arcs, and letting $N^k \subset N$ be the subset of nodes associated with (operations to be performed on) machine k, $k \epsilon Q = \{1, \cdots, q\}$, the disjunctive graph D defined by (α), (β), (γ), and (δ) has the following obvious properties:

Property 1. If 0 is the source and n the sink of G, then for every $j \epsilon N - \{0\} \cup \{n\}$ there is a path from 0 to j and a path from j to n in G. G is circuit-free (has no circuits).

Property 2. The set W of pairwise disjunctive arcs of D is such that $W = \bigcup_{k \epsilon Q} \{(i, j) \epsilon N^k \times N^k | i \neq j \text{ and there is no path connecting } i \text{ with } j \text{ in } G\}$.

$$\left\{\begin{matrix}(i, j) \text{ and } (r, s) \text{ form} \\ \text{a disjunctive pair}\end{matrix}\right\} \Leftrightarrow \left\{\begin{matrix}(i, j) \epsilon W,\ (r, s) \epsilon W \\ \text{and } r = j,\ s = i\end{matrix}\right\}. \tag{2}$$

For any $(i, j) \epsilon W$, the arc (j, i) is called the *complement* of (i, j), and the act of replacing (i, j) by (j, i) will be referred to as that of *complementing* (i, j).

A subset of W containing at most one arc of each disjunctive pair is called a *selection.* A selection containing exactly one arc of each disjunctive pair is *complete.* In this paper we consider only complete selections, hence we shall call them simply selections. Let

$$\mathcal{S} = \{S_1, \cdots, S_\sigma\} \tag{3}$$

be the set of all (complete) selections. Obviously, if $p = \frac{1}{2}|W|$ is the number of disjunctive pairs of arcs, then $\sigma = |\mathcal{S}| = 2^p$.

Each selection $S_h \epsilon \mathcal{S}$ generates a (conjunctive) graph of the form

$$G_h = (N, Z \cup S_h) = (N, Z_h). \tag{4}$$

A longest path from source to sink in G_h (if it exists, i.e., if G_h is circuit-free) is called a *critical* path in G_h.

Let

$$\mathcal{G} = \{G_1, \cdots, G_\sigma\}, \tag{5}$$

and

$$\mathcal{G}' = \{G_h \epsilon \mathcal{G} | G_h \text{ has no circuits}\}. \tag{6}$$

A critical path in $G_k \epsilon \mathcal{G}'$ is called a *minimaximal* path in D, and the associated selection S_k is called *optimal*, if, denoting by v_h the length of a critical path in G_h,

$$v_k = \min_{G_h \epsilon \mathcal{G}'} v_h. \tag{7}$$

The machine sequencing problem is then equivalent to:

Problem P.[10, 11] Find an optimal selection and an associated minimaximal path in the disjunctive graph D defined by (α), (β), (γ), and (δ).

In reference 11 we have used a specialized version of Benders's partitioning procedure to solve P by repeatedly solving two subproblems: a zero-one integer program and a critical path problem.

In the present paper, we describe an implicit enumeration procedure (of the type proposed in reference 12) which solves P by generating a sequence of circuit-free graphs $G_h \epsilon \mathcal{G}'$, and solving a slightly amended critical path problem for each G_h in the sequence. No integer program is to be solved. Each graph G_h is obtained from some previously generated term of the sequence by complementing one disjunctive arc. At each stage, some of the disjunctive arcs are fixed, while others are free (to be considered for being complemented); but the only candidates that actually need to be considered are the free arcs on a critical path of the current graph G_h. An easily obtained evaluation of the candidates is then used to direct the search so as to go from the current graph to the 'best' one that can be obtained by complementing one free disjunctive arc. At each stage the shortest critical path found so far provides a current upper bound for the search, whereas a critical path in the partial graph containing only the fixed arcs yields a current lower bound, which is tested against the current upper bound.

While computational experience is not yet available for determining the size of problems this method can solve in a reasonable time, the following features seem to be worth mentioning:

(a) The procedure can start with any feasible sequence (for instance, one actually used in production, or generated by some heuristics), and gradually improve it. Thus one can stop short of the optimum, with a reasonably 'good' feasible sequence.

(b) Storage requirements are limited to the data pertinent to the current node of the search tree.

At the beginning of this section, we have introduced the machine sequencing problem in its simplest possible form. However, the model we are using, i.e., the disjunctive graph defined by (α), (β), (γ), and (δ) can accommodate some more general situations than the case presented under (a), (b), (c), and (d) above.

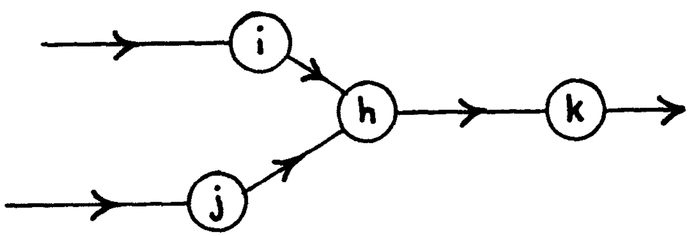

Figure 2

First, the model can handle items resulting from the assembly of other items; $G=(N, Z)$ will then contain a construction of the form shown in Fig. 2, where i and j are the last operations to be performed on two different items, h is the operation consisting of their assembly, and k is the first operation on a third item, resulting from the assembly of the first two.

Second, the operations pertaining to a given item may be only partially ordered. If 5 operations have to be performed on an item and only the following precedence relations are prescribed: $1<2$, $1<3$, $2<4$, $3<5$, $4<5$ (where $i<j$ means 'i has to precede j'), this can be translated by the set of conjunctive and disjunctive arcs shown in Fig. 3.

Third, set-up time for each operation may be taken into account by simply letting d_{ij} to be the completion time for operation i plus the set-up time for operation j.

Fourth, the same model can be used for sequencing lots of items rather than individual items. In this case d_{ij} is to be interpreted as the required lead time for operation i in relation to operation j.

However, this model does not include the case when a given operation is not related to an individual machine, but to a set of identical machines (i.e., can be performed on any machine in that set). This requires a generalization of the model, which is left to another paper.[13]

Two related, but different procedures for solving the model discussed in this paper are described in references 14 and 15.

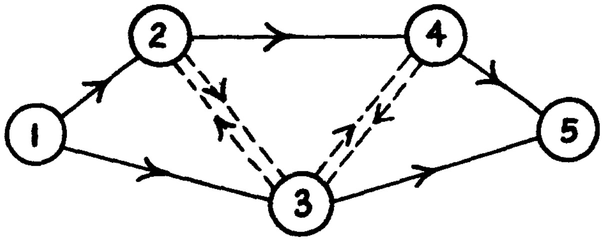

Figure 3

PROPERTIES OF THE DISJUNCTIVE GRAPH

CONSIDER THE disjunctive graph D defined by (α), (β), (γ), and (δ), and the related family of (conjunctive) graphs $\mathcal{G}'$ defined by (4), (5), and (6).

PROPOSITION 1. *For any $G_h \epsilon \mathcal{G}'$ and any arc $(i, j) \epsilon S_h$, there is a path in G_h from source to sink, containing (i, j).*

Proof. By Property 1, there is a path in G (hence in G_h), say $T(0, i)$, from 0 to i, and a path, say $T(j, n)$, from j to n. Hence the arcs in the set $T(0, i) \cup \{(i, j)\} \cup T(j, n)$ form a path in G_h from 0 to n.

We shall now make the following

ASSUMPTION. *For any $G_h \epsilon \mathcal{G}$, if (i, j) $\epsilon Z_h = Z \cup S_h$, then (i, j) is the unique shortest path from i to j.*

In other words, if there is a path $(i, h), (h, k), \cdots, (r, s), (s, j)$ from i to j in G_h, other than (i, j), then

$$d_{ij} < d_{ih} + d_{hk} + \cdots + d_{rs} + d_{sj}. \tag{8}$$

It follows from the meaning attributed to the numbers d_{ij} that this assumption is realistic for any disjunctive graph representing a machine sequencing problem.

PROPOSITION 2. *Let C_h be a critical path in $G_h \epsilon \mathcal{G}'$. Any graph G_k obtained from G_h by complementing one arc $(i, j) \epsilon S_h \cap C_h$ is circuit-free.*

Proof. $G_h \epsilon \mathcal{G}'$ is circuit-free by the definition of $\mathcal{G}'$. If complementing the arc (i, j) creates a circuit, then G_h contains a path from i to j, other than (i, j). However, since $(i, j) \epsilon C_h$, (i, j) is a longest path from i to j; also, according to the assumption made above, (i, j) is the unique shortest path from i to j in G_h. Hence (i, j) is the only path from i to j in G_h, and its complementing cannot create a circuit.

PROPOSITION 3. *Let C_h be a critical path in $G_h \epsilon \mathcal{G}'$. If there exists a graph $G_p = (N, Z \cup S_p) \epsilon \mathcal{G}'$ with a critical path C_p shorter than C_h, then the selection S_p contains the complement (j, i) of at least one arc $(i, j) \epsilon S_h \cap C_h$.*

Proof. If S_p does not contain the complement of any arc $(i, j) \epsilon S_h \cap C_h$, then $Z_p = Z \cup S_p$ contains C_h and a longest path in G_p cannot be shorter than C_h, which contradicts the assumption.

For any $G_h \epsilon \mathcal{G}$ and any node $j \epsilon N$, let

$$\mathcal{P}_h(j) = \{i \epsilon N | (i, j) \epsilon Z_h\} \tag{9}$$

and

$$\mathcal{S}_h(j) = \{i \epsilon N | (j, i) \epsilon Z_h\} \tag{10}$$

be the set of predecessors, and the set of successors, respectively, of node j in G_h. Further, for $G_h \epsilon \mathcal{G}'$, let $v_h(i, j)$ denote the length of a longest path from i to j, with the convention that $v_h(i, i) = 0$. It is well known (and obvious) that, for any node $j \epsilon N - \{0\}$,

$$v_h(0, j) = \max_{i \epsilon \mathcal{P}_h(j)} \{v_h(0, i) + d_{ij}\}, \tag{11}$$

and, for any $j \epsilon N - \{n\}$,

$$v_h(j, n) = \max_{i \epsilon \mathcal{S}_h(j)} \{d_{ji} + v_h(i, n)\}. \tag{12}$$

Let i_p and i_s be the nodes for which the right-hand sides of (11) and (12) attain their respective maxima. Further, if $\mathcal{P}_h(j) - \{i_p\} \neq \phi$, let

$$v_h'(0, j) = \max_{i \epsilon \mathcal{P}_h(j) - \{i_p\}} \{v_h(0, i) + d_{ij}\}, \tag{13}$$

and, if $\mathcal{S}_h(j) - i_s \neq \phi$, let

$$v_h'(j, n) = \max_{i \epsilon \mathcal{S}_h(j) - \{i_s\}} \{d_{ji} + v_h(i, n)\}. \tag{14}$$

In other words, let $v_h'(0, j)$ be the length of the longest path from 0 to j that does not contain the arc (i_p, j), and $v_h'(j, n)$ be the length of the longest path from j to n that does not contain the arc (j, i_s).

Finally, whenever $v_h'(0, j)$ and/or $v_h'(j, n)$ is defined, let

$$\delta_h(0, j) = v_h(0, j) - v_h'(0, j) \quad \text{and/or} \quad \delta_h(j, n) = v_h(j, n) - v_h'(j, n). \tag{15}$$

PROPOSITION 4. *The value*

$$\Delta_h(i, j) = d_{ij} + d_{ji} - \delta_h(0, j) - \delta_h(i, n) \tag{16}$$

is well-defined for all $(i, j) \epsilon S_h$.

Proof. $\Delta_h(i, j)$ is well-defined if $v_h'(0, j)$ and $v_h'(i, n)$ are well-defined, i.e., if j has more than one predecessor and i has more than one successor in G_h. But this is always the case, since (a) each node $j \epsilon N - \{0\} \cup \{n\}$ has a predecessor and a successor in $G = (N, Z)$ (Property 1 of D); (b) i is a predecessor of j (and j is a successor of i) in G_h, but not in G; and hence, in G_h, j has at least one predecessor other than i and i has at least one successor other than j.

PROPOSITION 5. *Let* $G_h \epsilon \mathcal{G}'$ *and let* G_k *be the graph obtained from* G_h *by complementing the arc* $(i, j) \epsilon C_h$, *where* C_h *is a longest path in* G_h. *Then*

$$\begin{aligned} v_k(0, n) &= v_h(0, n) + \Delta_h(i, j) \quad &\text{if} \quad \Delta_h(i, j) > 0, \\ v_h(0, n) \geqq v_k(0, n) &\geqq v_h(0, n) + \Delta_h(i, j) \quad &\text{if} \quad \Delta_h(i, j) \leqq 0. \end{aligned} \tag{17}$$

Proof. It is easy to see that $v_h(0, n) + \Delta_h(i, j)$ is the length of a longest path through (j, i), i.e., containing (j, i), in G_k. Indeed,

$$\begin{aligned} v_h(0, n) + \Delta_h(i, j) &= v_h'(0, j) + d_{ji} + v_h'(k, n) \\ &= v_k(0, j) + d_{ji} + v_k(i, n). \end{aligned} \tag{18}$$

But a longest path in G_k can be longer than a longest path C_h in G_h only if it contains (j, i), the sole arc of G_k that is not an arc of G_h. Hence, if $\Delta_h(i, j) > 0$, i.e., if a longest path through (j, i) in G_k is longer than C_h, then it is a longest path in G_k and its length $v_k(0, n)$ is given by the equality in (17). If, however, $\Delta_h(i, j) \leqq 0$, i.e., if a longest path through (j, i) in G_k is not longer than C_h, then the length $v_h(0, n)$ of C_h is an upper bound, and the length of a longest path through (j, i) in G_k is a lower bound, on the length $v_k(0, n)$ of a longest path in G_k—hence the inequalities in (17).

THE ALGORITHM

WE SHALL define a *direction* for each disjunctive pair of arcs in W by calling the arc (i, j) *normal* if $i<j$, and *reverse* if $i>j$. W^+ and W^- will denote the set of all normal arcs and the set of all reverse arcs respectively. Obviously, $W^+ \cup W^- = W$, $W^+ \cap W^- = \phi$, $W^+ \epsilon \mathcal{S}$, $W^- \epsilon \mathcal{S}$, and it is easy to see that the graphs $(N, Z \cup W^+)$ and $(N, Z \cup W^-)$ have no circuits.

The algorithm described below is illustrated by a numerical example in the next section. Starting with the graph $(N, Z \cup W^+)$ in which all disjunctive arcs are normal, we generate a sequence of graphs $G_h=(N, Z\cup S_h)\epsilon\mathcal{G}'$. Each G_h is obtained from some preceding term of the sequence by a forward step consisting of complementing one normal arc. Whenever for some graph G_k we can make sure that further complementing of normal arcs cannot bring any improvement, we abandon G_k and backtrack to the graph G_h from which G_k was generated. It is desirable for bookkeeping purposes to associate with this sequence an arborescence (rooted tree) A, a node of A being associated with each graph G_h, and an arc of A with each pair of graphs G_h, G_k such that G_k is obtained from G_h. Since G_k differs from G_h by exactly one disjunctive arc, say $(i, j)\epsilon S_h$, $(j, i)\epsilon S_k$, the arc (G_h, G_k) of A will also be associated with the disjunctive arc $(j, i)\epsilon S_k$.

Whenever a new graph G_h is obtained by complementing a normal arc (i, j) of some previously generated graph, the reverse arc (j, i) is temporarily *fixed* in G_h, in the sense that it cannot be complemented in any of the descendants G_p of G_h in A. (G_p is a *descendant* of G_h, and G_h is an *ancestor* of G_p, if there is a path from G_h to G_p in A.) On the other hand, whenever we backtrack to G_h from some successor G_k, we fix in G_h the normal arc (r, s), whose complementing had generated G_k. Thus, for each graph G_h generated under the procedure, a subset $F_h\subset S_h$ of disjunctive arcs is fixed. The reverse arcs of F_h are those associated with the path in A from the root to G_h, whereas the normal arcs of F_h are those associated with abandoned arcs of A, incident out of G_h or out of some ancestor of G_h in A. The arcs that are currently not fixed, i.e., those in the set S_h-F_h, are termed *free*. Of special importance at each stage is the set of *candidates*,

$$B_h=(S_h-F_h)\cap C_h, \tag{19}$$

i.e., the set of free arcs on a critical path C_h. (If G_h has more than one critical path, any one of them can be used to define B_h.) As it will be shown later, the only successors of G_h in A that may have to be generated under this procedure are those obtained by complementing an arc $(i, j)\epsilon B_h$. Also, each such successor is circuit-free.

The numbers $\Delta_h(i, j)$ defined by (16) evaluate the effect of complementing (i, j) on the length of a critical path in the resulting graph; hence, they can be used as a criterion for choosing among the candidates.

Further, at each stage we have a ceiling or current upper bound v^* on the length of a minimaximal path in D, given by the length of the shortest critical path found so far. Denoting by

$$G(F_h)=(N, Z\cup F_h) \tag{20}$$

the graph formed by dropping from G_h all currently free disjunctive arcs, and by $v(F_h)$ the length of a critical path in $G(F_h)$, it is obvious that whenever

$$v(F_h) \geqq v^* \tag{21}$$

the graph G_h with all its potential descendants in A can be abandoned.

We start with $G_1 = (N, Z \cup W^+)$, $F_1 = \phi$, i.e., all arcs normal and free, and $v^* = \infty$. We associate with G_1 the root of the search tree A.

A typical iteration of the algorithm may consist of the following steps. Let $G_r = (N, Z \cup S_r) = (N, Z_r)$ be the current graph and F_r the current set of fixed disjunctive arcs.

1. Test step. Compute the length $v(F_r)$ of a longest path in $G(F_r) = (N, Z \cup F_r)$. If $v(F_r) \geqq v^*$, backtrack (go to step 4). Otherwise go to step 2.

2. Evaluation step. For each $j \epsilon N$, compute $v_r(0, j)$ as defined by (11). If $v_r(0, n) < v^*$, set the new ceiling at $v^* = v_r(0, n)$.

Identify a critical path C_r and the set B_r of candidates, defined by (19). If there are several critical paths in G_r, choose any one, but only one, to define B_r.

If $B_r = \phi$, backtrack (go to step 4). Otherwise, for each $j \epsilon N$ such that $(i, j) \epsilon B_r$, compute $v_r'(0, j)$ as defined by (13), and for each $j \epsilon N$ such that $(j, i) \epsilon B_r$ compute $v_r(j, n)$ and $v_r'(j, n)$ as defined by (12) and (14). Then compute $\Delta_r(i, j)$ for each $(i, j) \epsilon B_r$ and go to step 3.

3. Forward step. Choose $(i, j) \epsilon B_r$ such that

$$\Delta_r(i, j) = \min_{(h,k) \epsilon B_r} \Delta_r(h, k) \tag{22}$$

[in case of ties, choose any arc satisfying (22)], and generate a new graph $G_s = (N, Z_s)$ by complementing the (normal) arc (i, j) and fixing the (reverse) arc (j, i), i.e., by letting

$$Z_s = [Z_r - \{(i, j)\}] \cup \{(j, i)\} \tag{23}$$

and

$$F_s = F_r \cup \{(j, i)\}. \tag{24}$$

Accordingly, add to the search tree A a new node G_s and a new arc (G_r, G_s), associated with the (reverse) arc (j, i) of the disjunctive graph D.

Then go to step 1.

4. Backtracking step. Backtrack to the predecessor G_p of G_r in A. If G_r has no predecessor, i.e., if we are instructed to backtrack from the root of A, the algorithm terminates: the selection S_k associated with the current v^* is optimal and a longest path in G_k is minimaximal in D.

Otherwise, let (j, i) be the (reverse) arc of D associated with the backtracking arc (G_p, G_r) of A. Then drop all data referring to G_r, and update the data for G_p, by removing the (normal) arc (i, j) from the set

B_p and introducing it into F_p, i.e., by replacing B_p and F_p by $B_p-\{(i, j)\}$ and $F_p \cup \{(i, j)\}$ respectively.

Then go to step 3.

PROPOSITION 6. *The algorithm consisting of steps 1, 2, 3, and 4 above finds a minimaximal path in D in a finite number of steps.*

Proof. By Propositions 1 and 2, each graph G_r generated under the procedure is circuit-free and has a critical path. Hence, and by Proposition 4, the steps of the algorithm are well-defined. (This is not true for more general disjunctive graphs, for which Proposition 1 does not hold: in such cases the procedure could generate a graph having no path from source to sink—which would make the steps of the algorithm ill-defined.) Further, since $\mathcal{G}'$ is a finite set, all we have to show is that (a) all elements of $\mathcal{G}'$ are explicitly or implicitly examined, and (b) no element is examined twice, i.e., the algorithm cannot cycle.

(a) A complete enumeration of the elements of $\mathcal{G}$ can be represented by means of a rooted tree T generated under the same rules as our search tree A, except that the successors of G_r in T are *all* graphs obtainable from G_r by complementing a free arc. Obviously, A is a subtree of T.

A node G_r of A (with all its descendants in T) is abandoned under the algorithm when Test 1 is failed, or when all successors of G_r obtainable by complementing a free arc on a critical path, have been examined. In the first case, no descendant of G_r in T can have a shorter critical path than the current best one, since all fixed arcs of G_h are fixed in all its descendants. In the second case, the same conclusion follows from Proposition 3.

(b) For each node G_r of A (and of T), and for all descendants of G_r in T, the set of fixed arcs contains all reverse arcs associated with the path from the root of A (of T) to G_r. Let us denote this set of arcs by R. After backtracking from G_r, the set of fixed arcs for the current node of A (which is either an ancestor of G_r in A, or a descendant of an ancestor) always contains the complement of at least one arc in R, namely the one associated with the last backtracking on the path from the root to G_r. Hence, after backtracking from G_r, the current node can never become identical with either G_r, or any of its descendants in T.

This completes the proof of Proposition 6.

There are two computational remarks we would like to make:

1. The values $v_r'(0, j)$ and $v_r'(j, n)$ can be computed at the same time with $v_r(0, j)$ and $v_r(j, n)$; thus the evaluation step involves little more than an ordinary critical path computation.

2. Storage requirements are minimal: besides the input data (list of nodes, list of conjunctive and disjunctive arcs with their respective lengths), all one needs is a full characterization of the current graph G_h in terms of the state of its disjunctive arcs (normal or reverse, free or fixed), including

the order in which the fixing has occurred (this is necessary in order to find the predecessor of G_h in A in case one has to backtrack).

ILLUSTRATION

We now illustrate the algorithm by solving a small numerical machine sequencing problem (example 1 of reference 11):

TABLE I

Machine \ Item	1	2	3
1	7	10	9
2	11	5	8

Three items (lots of items) are to be processed on two machines: first on machine 1, then on machine 2. The times required for processing each item on each machine are shown in Table I, while Table II indexes the operations. The associated disjunctive graph is shown in Fig. 4.

The following is a brief description of the successive iterations, also illustrated in Figs. 5, 6, and 7.

The first three graphs $G_h \epsilon \mathcal{G}$ are shown in Fig. 5. Critical paths are marked by thick lines (full or broken). Heavy, solid arrows indicate fixed arcs. The numbers in the four boxes for each node j are, from left to right, $v_h'(0, j)$, $v_h(0, j)$, $v_h(0, n) - v_h(j, n)$, and $v_h(0, n) - v_h'(j, n)$. Thus the absolute difference between the numbers in the first two boxes is $\delta_h(0, j)$, and that between the numbers in the last two boxes is $\delta_h(j, n)$.

TABLE II

Operation	1	2	3	4	5	6
Item	1	1	2	2	3	3
Machine	1	2	1	2	1	2

The other graphs G_h or $G_h(F_h)$ generated are pictured in Fig. 6, while Fig. 7 shows the arborescence A associated with the problem.

The graph G_3 is optimal. There are two minimaximal paths of length 31, which is the minimum total time for processing the three items. The optimal sequence of operations is given by G_3.

1. We start with $G_1 = (X, Z \cup W^+)$, $F_1 = \phi$, $v^* = \infty$.
   Test: $v(F_1) = 18 < \infty$.
   Evaluation: $v_1 = v_1(0, 7) = 34$, set $v^* = 34$; $B_1 = \{(1, 3), (3, 5)\}$.
   $$\Delta_1(1, 3) = d_{13} + d_{31} - \delta_1(0, 3) - \delta_1(1, 7) = 7 + 10 - 7 - 3 = 7.$$
   $$\Delta_1(3, 5) = d_{35} + d_{53} - \delta_1(0, 5) - \delta_1(3, 7) = 10 + 9 - 10 - 4 = 5.$$
   Forward: Choose (3, 5) and generate G_2 by fixing (5, 3).

2. $F_2 = \{(5, 3)\}$.
Test: $v(F_2) = 24 < 34$.
Evaluation: $v_2 = 39$; $B_2 = \{(1, 5), (4, 6)\}$.
$\Delta_2(1, 5) = 7+9-7-8 = 1$; $\Delta_2(4, 6) = 5+8-13-8 = -8$.
Forward: Choose (4, 6) and generate G_3 by fixing (6, 4).
3. $F_3 = \{(5, 3), (6, 4)\}$.
Test: $v(F_3) = 24 < 34$.
Evaluation: $v_3 = 31$, set $v^* = 31$; $B_3 = \{(2, 6)\}$ (of the two critical paths, we use the one shown by a broken line in Fig. 5).
$\Delta_3(2, 6) = 11+8-2-8 = 9$.
Forward: Choose (2, 6); generate G_4.
4. $F_4 = \{(5, 3), (6, 4), (6, 2)\}$.
Test: $v(F_4) = 28 < 31$.

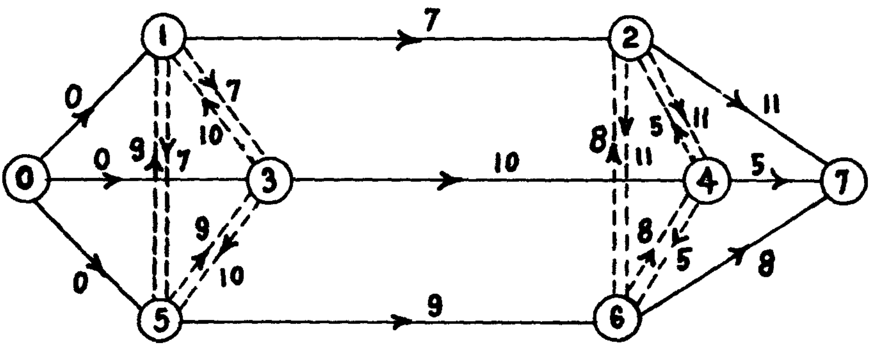

Figure 4

Evaluation: $v_4 = 40$; $B_4 = \{(1, 5), (2, 4)\}$.
$\Delta_4(1, 5) = 7+9-7-17 = -8$, $\Delta_4(2, 4) = 11+5-9-5 = 2$.
Forward: Choose (1, 5); generate G_5.
5. $F_5 = \{(5, 3), (6, 4), (6, 2), (5, 1)\}$.
Test: $v(F_5) = 28 < 31$.
Evaluation: $v_5 = 33$; $B_5 = \{(2, 4)\}$.
$\Delta_5(2, 4) = 11+5-2-5 = 9$.
Forward: Generate G_6 by fixing (4, 2).
6. $F_6 = \{(5, 3), (6, 4), (6, 2), (5, 1), (4, 2)\}$.
Test: $v(F_6) = 35 > 31$.
Backtrack: Introduce (2, 4) into F_5. Then $B_5 = \phi$, hence again:
Backtrack: Introduce (1, 5) into F_4. Then $B_4 = \{(2, 4)\}$.
Forward: Generate G_7 by fixing (4, 2).
7. $F_7 = \{(5, 3), (6, 4), (6, 2), (1, 5), (4, 2)\}$.
Test: $v(F_7) = 42 > 31$.
Backtrack: Introduce (2, 4) into F_4. Then $B_4 = \phi$, hence:
Backtrack: Introduce (2, 6) into F_3. Then $B_3 = \phi$, hence:

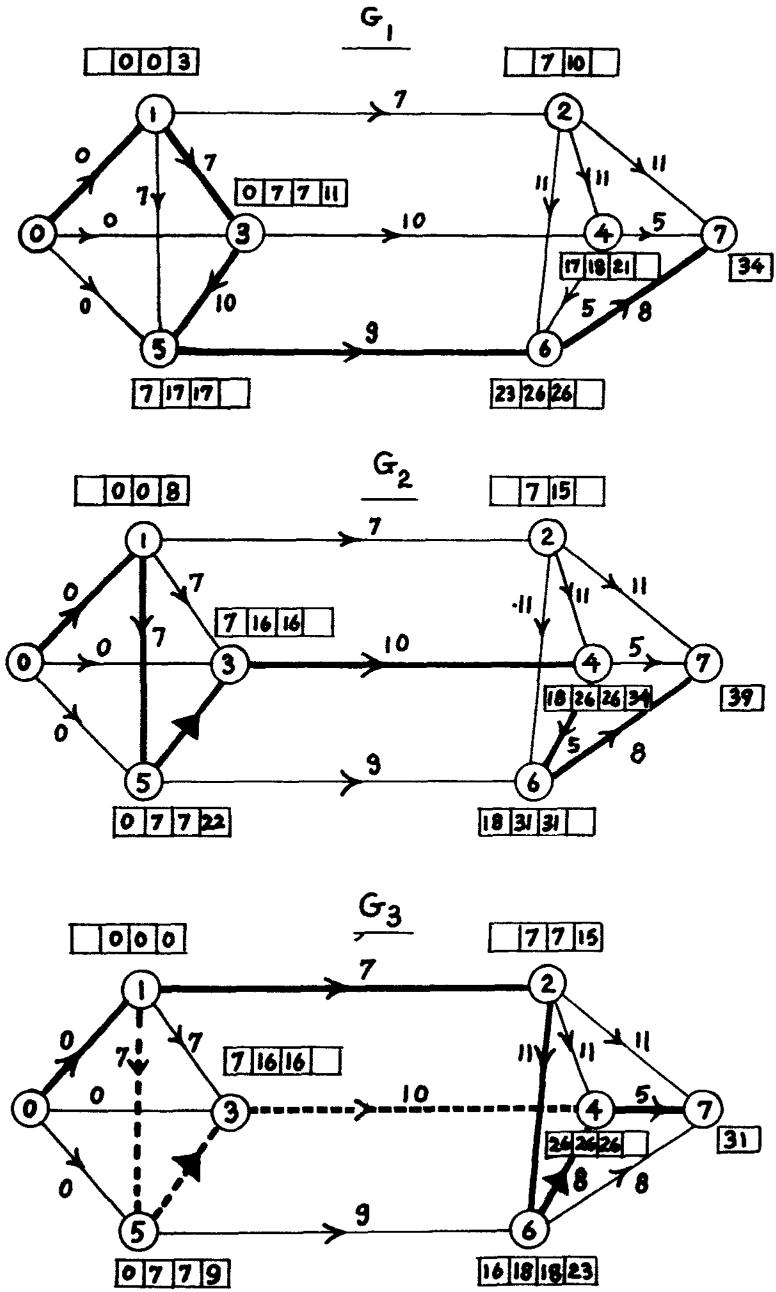

Figure 5

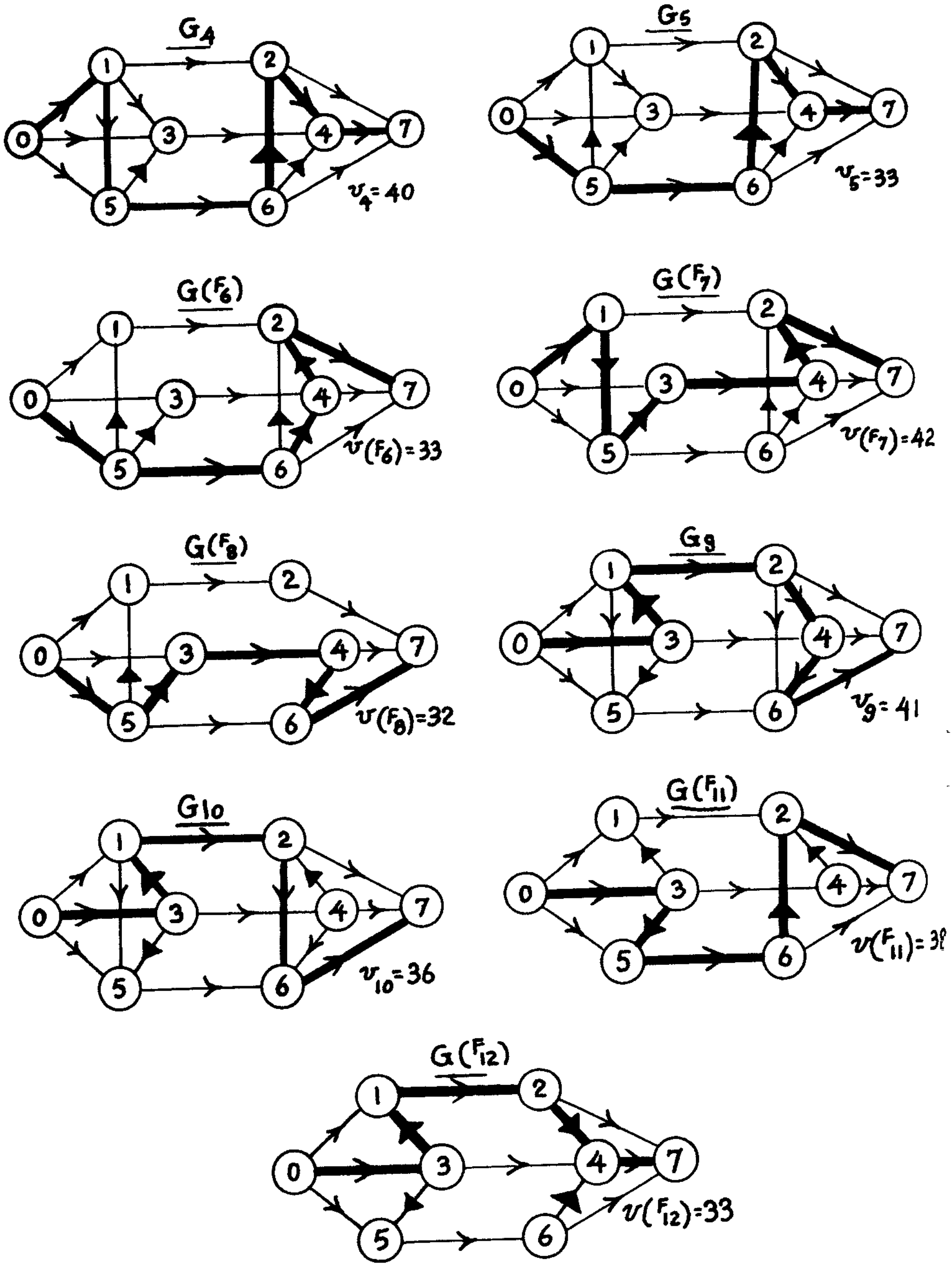
G4
v4= 40
G5
v5=33
G(F6)
v(F6)=33
G(F7)
v(F7)=42
G(F8)
v(F8)=32
G9
v9= 41
G10
v10=36
G(F11)
G(F12)
v(F12)=33

Figure 6

Backtrack: Introduce (4, 6) into F_2. Then $B_2 = \{(1, 5)\}$.
Forward: Generate G_8 by fixing (5, 1).
8. $F_8 = \{(5, 3), (4, 6), (5, 1)\}$.

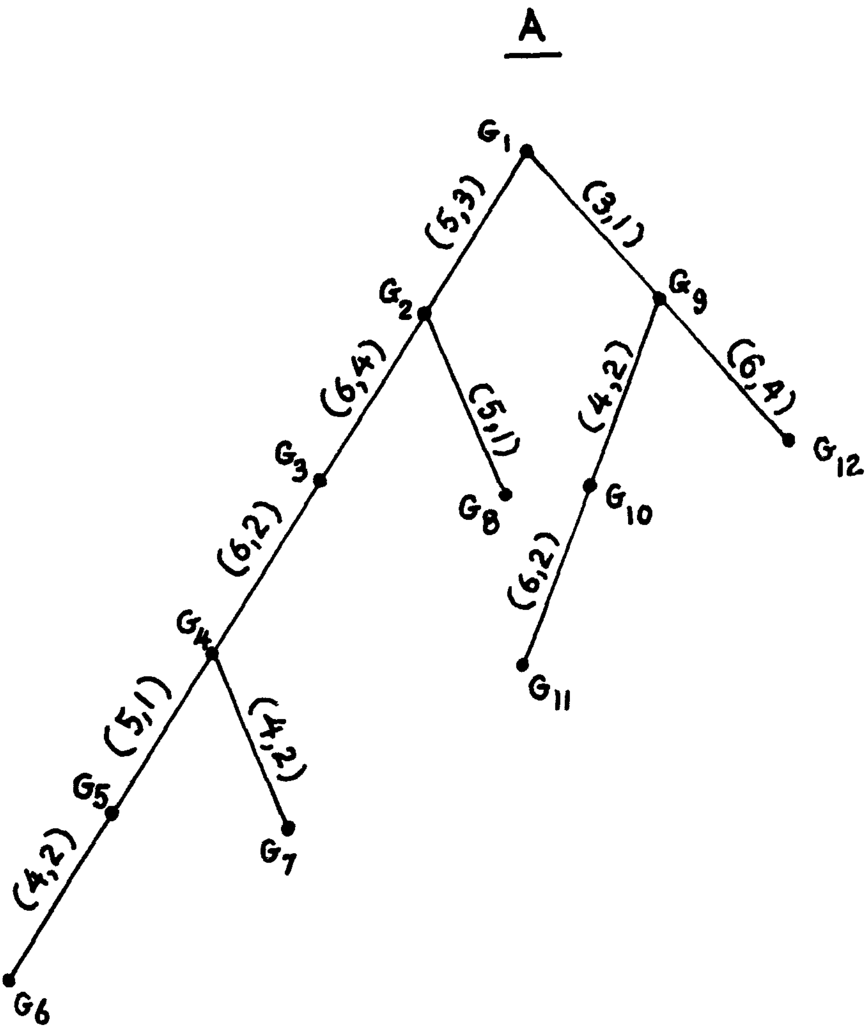

Figure 7

Test: $v(F_8) = 32 > 31$.
Backtrack: Introduce (1, 5) into F_2. Then $B_2 = \phi$, hence:
Backtrack: Introduce (3, 5) into F_1. Then $B_1 = \{(1, 3)\}$.
Forward: Generate G_9 by fixing (3, 1).

9. $F_9 = \{(3, 5), (3, 1)\}$.
Test: $v(F_9) = 28 < 31$.
Evaluation: $v_9 = 41$; $B_9 = \{(2, 4), (4, 6)\}$.
$\Delta_9(2, 4) = 11+5-18-5 = -7$; $\Delta_9(4, 6) = 5+8-7-8 = -2$.
Forward: Choose (2, 4), generate G_{10}.
10. $F_{10} = \{(3, 5), (3, 1), (4, 2)\}$.
Test: $v(F_{10}) = 28 < 31$.
Evaluation: $v_{10} = 36$; $B_{10} = \{(2, 6)\}$.
$\Delta_{10}(2, 6) = 11+8-2-8 = 9$.
Forward: Fix (6, 2) and generate G_{11}.
11. $F_{11} = \{(3, 5), (3, 1), (4, 2), (6, 2)\}$.
Test: $v(F_{11}) = 38 > 31$.
Backtrack: Introduce (2, 6) into F_{10}. Then $B_{10} = \phi$, hence:
Backtrack: Introduce (2, 4) into F_9. Then $B_9 = \{(4, 6)\}$.
Forward: Generate G_{12} by fixing (6, 4).
12. $F_{12} = \{(3, 5), (3, 1), (2, 4), (6, 4)\}$.
Test: $v(F_{12}) = 33 > 31$.
Backtrack: Introduce (4, 6) into F_9. Then $B_9 = \phi$, hence:
Backtrack: Introduce (1, 3) into F_1. Then $B_1 = \phi$, hence:
Backtrack: End.

ACKNOWLEDGMENTS

I WISH to thank a referee for his useful comments on an earlier version of this paper. This report was partially supported by IBM's New York Scientific Center. It was prepared as part of the activities of the Management Sciences Research Group, Carnegie-Mellon University, under a contract with the US Office of Naval Research.

REFERENCES

1. S. M. JOHNSON, "Optimal Two- and Three-Stage Production Schedules with Set-Up Times Included," *Naval Res. Log. Quart.* **1,** 61–68 (1954).
2. F. H. BOWMAN, "The Schedule-Sequencing Problem," *Opns. Res.* **7,** 621–624 (1959).
3. H. WAGNER, "An Integer Programming Model for Machine Scheduling," *Naval Res. Log. Quart.* **6,** No. 2 (1959).
4. G. B. DANTZIG, "A Machine-Job Scheduling Model," *Management Sci.* **6,** 191-196 (1960).
5. A. S. MANNE, "On the Job-Shop Scheduling Problem," *Opns. Res.* **8,** 219–223 (1960).
6. B. GIFFLER AND G. L. THOMPSON, "Algorithms for Solving Production Scheduling Problems," *Opns. Res.* **8,** 487–503 (1960).
7. L. NÉMETI, "Das Reihenfolgeproblem in der Fertigungsprogrammierung und Linearplanung mit logischen Bedigungen," *Mathematika* **6**(29) (1964).
8. S. ROY AND B. SUSSMAN, "Les problèmes d'ordonnancement avec contraintes disjonctives," *SEMA*, Note D.S. No. 9 bis (décembre 1964).

9. R. W. CONWAY, W. L. MAXWELL, AND L. W. MILLER, *Theory of Scheduling*, Addison-Wesley, New York, 1964.
10. E. BALAS, "Finding a Minimaximal Path in a Disjunctive PERT Network," *Théorie des Graphes, Journées internationales d'étude, Rome, Juillet 1966*, Dunod, Paris—Gordon and Beach, pp. 21–30, New York, 1967.
11. ———, "Discrete Programming by the Filter Method," *Opns. Res.* **15**, 915–957 (1967).
12. ———, "An Additive Algorithm for Solving Linear Programs with Zero-One Variables," *Opns. Res.* **13**, 517–546 (1965).
13. ———, "Project Scheduling with Resource Constraints," *Proceedings of the NATO Conference on Applications of Mathematical Programming*, Cambridge (England), June 24–28, 1968.
14. J. F. RAIMOND, "An Algorithm for the Exact Solution of the Machine Scheduling Problem," IBM, New York Scientific Center Report No. 320-2930 (February 1968).
15. E. BALAS, "Machine Sequencing: Disjunctive Graphs and Degree-Constrained Subgraphs," IBM, New York Scientific Center Report No. 320-2971 (April 1969).