Operations Research

Publication details, including instructions for authors and subscription information:
http://pubsonline.informs.org

Discrete Programming by the Filter Method

Egon Balas,

To cite this article:
Egon Balas, (1967) Discrete Programming by the Filter Method. Operations Research 15(5):915-957. http://dx.doi.org/10.1287/opre.15.5.915

DISCRETE PROGRAMMING BY THE FILTER METHOD*

Egon Balas

Stanford University, Stanford, California

(Received June 21, 1966)

In this paper (section 1) a two-phase procedure, the filter method or accelerated additive algorithm, is proposed for solving linear programs with zero-one variables. In Phase I an auxiliary problem is constructed that, in Phase II, is used to 'filter' the solutions to which the tests of the additive algorithm are to be applied. The filter method is then extended (section 2) by J. F. BENDERS' partitioning procedure to the mixed-integer zero-one case, as well as to general integer and mixed-integer programs. Finally, a specialized version of this method is used (section 3) to tackle a general machine-sequencing model, formulated as the problem of finding a minimaximal path in a disjunctive graph.

IN REFERENCES 1, 2, and 3 a partial enumeration method (the additive algorithm) was proposed for solving linear programs with zero-one variables, and in reference 4 this method was extended to more general types of integer programming problems. The results contained in the present paper are a direct continuation of those in the above ones. However, this paper is self-contained and the reader will not be supposed to be acquainted with the additive algorithm. A brief note on the filter method and another one on its extension to the mixed-integer problem have been published.[5,6]

Discrete programming is at present one of the most rapidly developing areas of operations research. One good reason for this is the remarkably wide range of problems that can be translated by its models, especially by those with 0-1 variables (*see*, for instance, [7]). Several approaches have so far been tried for solving problems of this type, which have generated a number of methods. Most of them have been quoted in reference 3; thus here we shall confine ourselves to add some of the new developments that have come to our knowledge (*see* references 8–12, 39, and 42). For a comprehensive survey of the field, *see* reference 13. However, in spite of the impressive number of methods, the question of finding *efficient* procedures for the handling of really large problems is still open.

The partial enumeration approach of the additive algorithm continues to look promising. The algorithm itself has been implemented and tested in many places. B. BOUVIER AND G. MESSOUMIAN[14] at the University of

* The research underlying this paper was partially sponsored by INCEF, Bucharest, and partially by the International Computation Centre, Rome.

Grenoble and RAOUL J. FREEMAN[15] at the Rand Corporation have programmed two slightly modified versions of the additive algorithm and solved 25 and 17 test problems respectively with up to 30 variables and 25 constraints. Both testings have shown that (a) the limiting factor on the size of problems that can be solved by this method is computing time; (b) computing time tends to reduce if the number of constraints increases; (c) reasonably 'good' feasible solutions can be obtained by the method in a small fraction of the time required to obtain an optimal solution *and* the information that it is optimal. Further, some versions of the algorithm have been successfully used by C. C. PETERSEN[16,43] of Motorola, Inc., to solve capital budgeting problems involving up to 50 variables. More recently, BERNHARD FLEISCHMANN,[27] at the Deutsches Rechenzentrum in Darmstadt, has implemented the additive algorithm with some modifications and has solved a number of problems with up to 159 variables in reasonably good computing time on an IBM 7094. Two problems with 70 and 90 variables respectively could not be solved in due time, but several other problems with more than 100 variables were solved in less than 3 minutes. The largest among them, with 159 variables, took 2.45 minutes to be solved.

These experiments, and fruitful discussions and developments of the algorithm by P. BERTIER AND P. T. NGHIEM,[17] F. GLOVER,[9] F. GLOVER AND S. ZIONTS,[18] A. GEOFFRION,[19] AND OTHERS have strongly stimulated the author's further research. (For more recent discussions, *see also* references 38, 39, 40, 41, 47, 48, and 49. An improved restatement of the additive algorithm is to be found in reference 45).

1. THE FILTER METHOD OR ACCELERATED ADDITIVE ALGORITHM

Implicit Enumeration and the Solution-Graph of the Zero-One Problem

We shall consider the problem P, which consists of finding a vector $x=(x_1, \cdots, x_n)$ such that

$$cx=\min, \tag{1}$$

$$Ax \geqq b, \tag{2}$$

$$0 \leqq x_j \leqq 1, \qquad (j=1, \cdots, n) \tag{3}$$

$$x_j \text{ integer} \qquad (j=1, \cdots, n) \tag{4}$$

where $c=(c_j)$, $b=(b_i)$, $A=(a_{ij})$, $(i=1, \cdots, m;\ j=1, \cdots, n)$, are given. Let $\{1, \cdots, n\}=N$. Without loss of generality, we shall suppose that $c \geqq 0$ (if $c_j<0$ for some $j \epsilon N$, one substitutes $x'_j=1-x_j$). We shall also consider the 'continuous' correspondent P' of P, which consists of finding $x=(x_1, \cdots, x_n)$ subject to (1), (2), and (3).

We shall call a *solution* to P (to P') any n-dimensional vector x satisfying (3) and (4) [respectively (3)]. A solution to P (to P') satisfying (2) will be called *feasible*. A feasible solution to P (to P') minimizing cx will be called *optimal*.

The approach of the additive algorithm to the solution of P is that of partial enumeration. By *partial* or *implicit enumeration* we mean a procedure for systematically enumerating part of the solutions to P and examining them in such a way as to ensure that by enumerating a (relatively small) number of solutions, we have implicitly examined *all* elements of the solution set.

Of course, any such approach is bound to classify in one way or another the elements of the solution set. There are 2^n solutions to P. The set of these solutions can be partitioned in $n+1$ subsets, so that the kth subset ($k=0, 1, \cdots, n$) contains all solutions with exactly k components of x being equal to 1. Thus the first subset has one single element (the solution $x_j=0$, $\nexists j \epsilon N$), the second subset has n elements (all solutions of the form $x_{j_1}=1$, $x_j=0$, $\nexists j \neq j_1$, for $j_1=1, \cdots, n$), etc. Generally, the number of elements in the kth subset is given by the binomial coefficient

$$\binom{n}{k} = n!/k!(n-k)!.$$

The relations between the elements of the solution set can be described with the aid of a graph G. A node r of G is associated with each solution x^r, and an arc (r, s) is associated with each pair of solutions x^r, x^s, having the property that $x_j^s = 1$ for all $j \epsilon J_r$ and for exactly one $j \epsilon N - J_r$, where $J_r = \{j | x_j^r = 1\}$. Fig. 1a shows the solution-graph of a problem with $n=4$. The numbers assigned to each node i are the elements of J_i. (A graph of this type is used in reference 17).

The original additive algorithm constructs a tree in the solution-graph G. Starting from node 0, we choose an arc, say $(0, k)$ (i.e., a variable x_j to be assigned the value 1) according to a certain rule, thus reaching a new node k (i.e., a new solution). Whenever a new node s is attained, the corresponding solution is submitted to certain tests meant to discover whether a feasible solution 'better' than those already found (if there are such), can exist beyond node s. If the tests are passed, the procedure continues with the choice of a new arc, say (s, t). If not, then s is abandoned with all its descendants, and we backtrack to the last node visited before s, say r. The procedure is then continued from node r, but the rules for the construction of the tree are such that no descendant of any abandoned node will ever be visited. For instance, if node 4 in Fig. 1b is abandoned, then together with it half of all the nodes are excluded from any further investigation (14, 24, 34, 124, 134, 234, 1234). The efficiency of the algorithm obviously depends on the size of the solution-tree to be

constructed before an optimal feasible solution is found or the absence of a feasible solution is established.

One way of reducing the size of this solution-tree is that of sharpening the tests of the algorithm. Several proposals have been made in this direction in references 9, 15, 18, 27, and 41. They deserve to be explored.

However, here we shall follow another way, based on the idea of 'filtering' the nodes of the solution-graph G *before* submitting them to the tests of the additive algorithm. The role of the 'filter' will be played by an

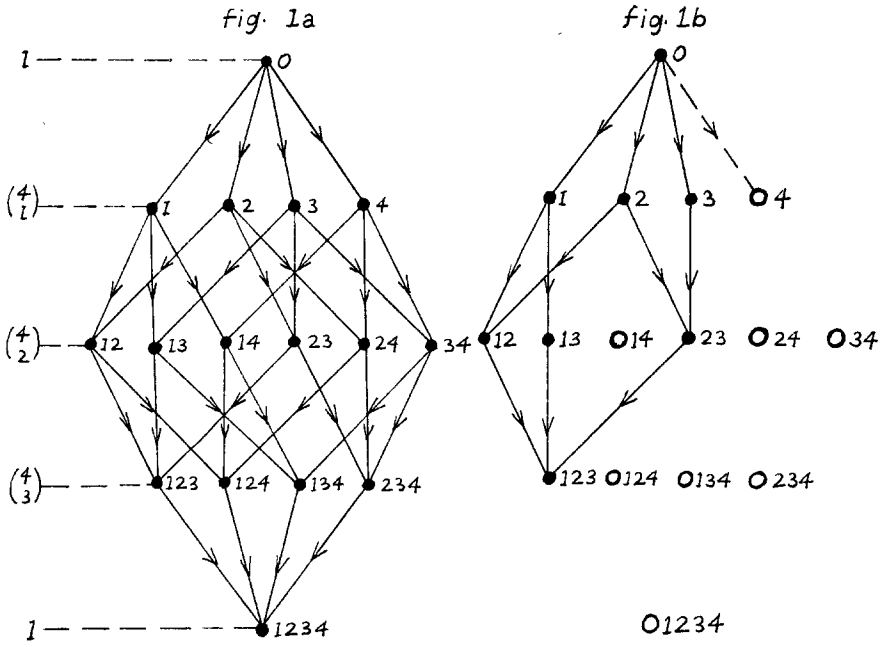

Figure 1

auxiliary 0-1 problem F, with the same solution-set (solution-graph) and objective function as P, but with a single constraint of type (2), such that $X(P) \subset X(F) \subset X'$, where $X(P)[X(F)]$ is the set of feasible solutions to P (to F), while $X' = \{x | cx \geqq c\bar{x},\ x_j = 0 \text{ or } 1,\ j = 1, \cdots, n\}$, $\bar{x}$ being an optimal solution to P'. A sequence of feasible solutions x^k to F will be generated ($k = 1, 2, \cdots$), such that the sequence of associated values cx^k is nondecreasing. This sequence corresponds to a (generally disconnected) subgraph of the solution-graph G. Each term of the sequence will be submitted to the tests of the additive algorithm, and if the tests are not passed, the associated node of G, together with all its descendants, will be excluded from further consideration. The first term of the sequence satisfying (2) (if such a term exists) will be an optimal solution to P.

The Filter

Let us consider the problem P' and its dual D'. If P' has a finite optimal solution, then D' also has one. Suppose this is the case, and let $\bar{x}$, $(\bar{u}, \bar{v})$ be a pair of optimal solutions, to P' and D' respectively, u_i and v_j being the dual variables associated with the ith constraint of (2) and the jth constraint (upper bound) of (3) respectively.

Now consider the filter-problem $F(\bar{u})$ which consists of finding $x=(x_1, \cdots, x_n)$ such that

$$cx = \min, \tag{1}$$

$$ax \geqq b_0, \tag{2'}$$

$$0 \leqq x_j \leqq 1, \qquad (j=1, \cdots, n) \tag{3}$$

$$x_j \text{ integer}, \qquad (j=1, \cdots, n) \tag{4}$$

where $a=\bar{u}A$, $b_0=\bar{u}b$. Let $F'(\bar{u})$ stand for the 'continuous' correspondent of $F(\bar{u})$, which consists of finding $x=(x_1, \cdots, x_n)$ subject to (1), (2'), and (3).

We observe that (2') is a very special case of the surrogate-constraint (s-constraint), defined by F. Glover,[9] as a nonnegative linear combination of the constraints (2) with at least one positive weight, and used with the tests of his multiphase-dual algorithm along with the other constraints of the problem. The rationale behind the idea of a surrogate-constraint is that to a certain extent it concentrates the restrictive properties of the individual constraints of (2); and the more it does so, the more useful it is.

As we shall see from theorem 1, our s-constraint (2'), obtained at a relatively high computational price (the solving of P'), has in return very useful properties justifying the central role assigned to the filter-problem $F(\bar{u})$ in the guidance of our search process.

Theorem 1. *If $\bar{x}$, $(\bar{u}, \bar{v})$ is a pair of optimal solutions to P' and D' respectively, then $\bar{x}$ is also an optimal solution to $F'(\bar{u})$ and $c\tilde{x} \geqq c\bar{x}$ for any feasible solution $\tilde{x}$ to $F(\bar{u})$. There exists at least one pair of optimal solutions $\hat{x}$, $(\hat{u}, \hat{v})$ to P' and D' respectively, such that $\hat{x}_j=1 \Rightarrow \bar{x}_j=1$ and $\hat{x}_j=0 \Rightarrow \bar{x}_j=0$ for any optimal solution $\bar{x}$ to $F'(\hat{u})$.*

Proof. (a) *First part of the theorem:*

P' and its dual D' can be written in the form

$$P' \begin{cases} cx = \min, \\ Ax \geqq b, \\ -x \geqq -e, \\ x \geqq 0, \end{cases} \qquad D' \begin{cases} ub - ve = \max, \\ uA - v \leqq c, \\ (u, v) \geqq 0, \end{cases}$$

where e is the n-dimensional vector $(1, \cdots, 1)$.

For any pair of optimal solutions $\bar{x}$, $(\bar{u}, \bar{v})$, to P' and D' respectively, we have

$$c\bar{x} = \bar{u}b - \bar{v}e.$$

Now, any feasible solution x to $F'(\bar{u})$, i.e., any $x \geqq 0$ verifying $\bar{u}Ax \geqq \bar{u}b$ and $-x \geqq -e$, also verifies the following nonnegative linear combination of these constraints:

$$\bar{u}Ax - \bar{v}x \geqq \bar{u}b - \bar{v}e.$$

On the other hand, substituting $u = \bar{u}$ and $v = \bar{v}$ in the constraint set of D', we obtain for any $x \geqq 0$

$$cx \geqq \bar{u}Ax - \bar{v}x.$$

Hence, for any feasible solution x to $F'(\bar{u})$ we have

$$cx \geqq \bar{u}b - \bar{v}e = c\bar{x},$$

and obviously $c\tilde{x} \geqq c\bar{x}$ for any feasible solution $\tilde{x}$ to $F(\bar{u})$.

(*b*) *Second part of the theorem:*

Let us denote by A_j the columns of A and by a_j the components of $a = \bar{u}A$.

The strong theorem on complementary slackness (corollary 2A, par. 2[20]) implies that there exists at least one pair $\hat{x}$, $(\hat{u}, \hat{v})$ of optimal solutions to P' and D' respectively, such that

$$\hat{x}_j = 1 \Rightarrow (\hat{u}A_j - \hat{v}_j = c_j,\ \hat{v}_j > 0) \Rightarrow a_j > c_j,$$

$$\hat{x}_j = 0 \Rightarrow (\hat{u}A_j - \hat{v}_j < c_j,\ \hat{v}_j = 0) \Rightarrow a_j < c_j,$$

$$0 < \hat{x}_j < 1 \Rightarrow (\hat{u}A_j - \hat{v}_j = c_j,\ \hat{v}_j = 0) \Rightarrow a_j = c_j.$$

Since $\hat{x}$ is a feasible (optimal) solution to $F'(\hat{u})$, it is obvious from the above that any optimal solution $\tilde{x}$ to $F'(\hat{u})$ can differ from $\hat{x}$ only by its components $\tilde{x}_j$ such that $0 < \tilde{x}_j < 1$.

Theorem 1 is thus proved.

As we see from theorem 1 if our search for an optimal solution to P is confined to the solution-set of our filter-problem, this will at once eliminate from consideration all solutions x such that $cx < c\bar{x}$. On the other hand, a vector $x = (x_j)$, $x_j = 0$ or 1, $j = 1, \cdots, n$, satisfying $cx \geqq c\bar{x}$, needs not necessarily be a feasible solution to $F(\bar{u})$; thus, part of the solutions x such that $cx \geqq c\bar{x}$ will also be eliminated by the filter.

Generating and Testing Solutions

What we need now is an implicit enumeration technique for systematically generating a sequence of feasible solutions x^k to $F(\bar{u})$, such that (α) the sequence of associated values cx^k is nondecreasing, and (β) when a certain value cx^s has been reached, then *all* feasible solutions to $F(\bar{u})$ such that $cx < cx^s$ have been explicitly or implicitly enumerated; so that, the first term of the sequence that is a feasible solution to P, is also optimal. This will be achieved through a procedure of the 'branch-and-bound' type

(*see* references 21, 22, 23, 17, 12, and 44), specialized for our purposes and combined with the tests of the additive algorithm.

In the following we shall write F and F' instead of $F(\bar{u})$ and $F'(\bar{u})$ whenever this is not confusing.

Let us first renumber the components of x_j according to the rule that, for any pair of indices $j_1 \epsilon N$, $j_2 \epsilon N$, we have $j_1 < j_2$ whenever one of the following three situations holds:

1. $a_{j_1} > 0, \quad a_{j_2} \leqq 0;$
2. $a_{j_1} > 0, \quad a_{j_2} > 0, \quad c_{j_1}/a_{j_1} < c_{j_2}/a_{j_2};$
3. $a_{j_1} \leqq 0, \quad a_{j_2} \leqq 0, \quad a_{j_1} > a_{j_2}.$

Situations 1, 2, and 3 are not exhaustive, so this rule does not uniquely define the indexing. Any indexing that satisfies the above rule, will be called *optimal.*

Our procedure will consist of systematically assigning values 0 or 1 to some components of x. A set of assignments

$$\psi_s = \{x_{j_1} = \delta_{j_1}^s, \cdots, x_{j_q} = \delta_{j_q}^s\}, \tag{5}$$

where δ_j^s $(j = j_1, \cdots, j_q)$ is one of the values 0 or 1, and $1 \leqq q \leqq n$, will be called a *pseudo-solution.* Let $N = N_s \cup \bar{N}_s$, where $\bar{N}_s = \{j_1, \cdots, j_q\}$ and N_s is the set of *free* indices, i.e., those corresponding to components of x not assigned a value by ψ_s (free components), and let F_s' denote the problem obtained from F' by introducing the additional constraints

$$x_j = \delta_j^s. \qquad (\times j \epsilon \bar{N}_s) \tag{6}$$

With each pseudo-solution ψ_s we associate the value $z_s = z(\psi_s) = c\bar{x}^s$, where $\bar{x}^s = (\bar{x}_1^s, \cdots, \bar{x}_n^s)$ is an optimal solution to F_s'. If ψ_s is such that the vector x^s defined by

$$x_j^s = \begin{cases} \delta_j^s, & (j \epsilon \bar{N}_s) \\ 0, & (j \epsilon N_s) \end{cases} \tag{7}$$

is an optimal solution to F_s', then x^s is also a feasible solution to F.

The procedure we shall describe generates a sequence of pseudo-solutions $\psi_1, \psi_2, \cdots$. We shall associate with this sequence a graph T. Each node r of T corresponds to a term ψ_r of the sequence and each arc (r, s) of T corresponds to a pair of terms ψ_r, ψ_s with the property that $\bar{N}_r \subset \bar{N}_s$, $\delta_j^s = \delta_j^r$, $\times j \epsilon \bar{N}_r$, and that no term ψ_t of the sequence exists, such that $\bar{N}_r \subset \bar{N}_t \subset \bar{N}_s$ and $\delta_j^s = \delta_j^t$, $\times j \epsilon \bar{N}_t$.

Relation (7) associates with each pseudo-solution ψ_s a solution x^r, and hence with each node of T, a node of G. Our procedure is such that T is a tree of a form similar to that shown in Fig. 2. The root of T, i.e., the starting term of our sequence, is the 'improper' pseudo-solution ψ_α defined by

$\bar{N}_\alpha = \phi$, with the associated value $z_\alpha = c\bar{x}$, where $\bar{x}$ is an optimal solution to P' (and hence to F' or, which is the same thing, to $F_\alpha{}'$).

Starting with the root of T, at each iteration a node s of T' is selected according to a choice rule to be explained below (*see* the Choice Step of the algorithm). The solution x^s associated by (7) with the selected pseudo-solution ψ_s, may be feasible or not for F. Let us suppose it is not. Then we apply a branching procedure (*see* the Branching Step), generating a set of pseudo-solutions ψ_{st} $(t=1, 2, \cdots)$ as follows. Let $N_s(t)=\{j_1, \cdots, j_t\}$

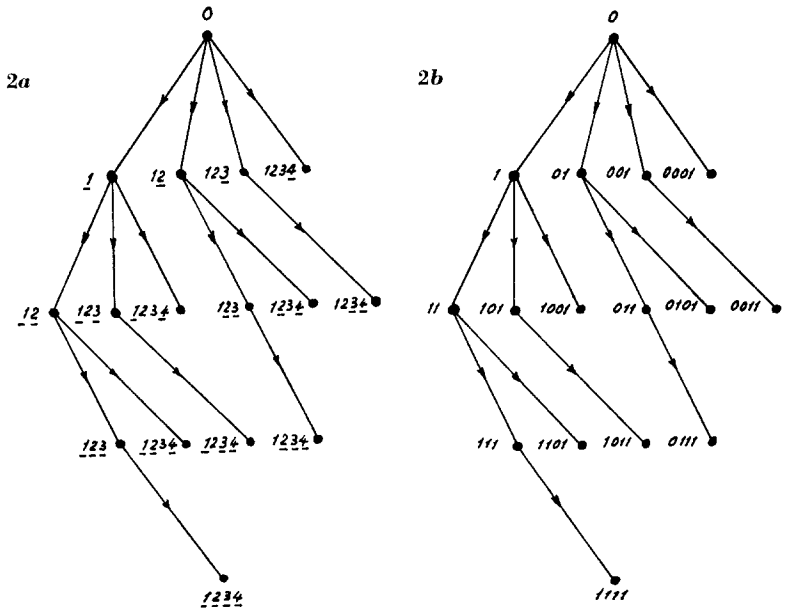

Figure 2

be the set of the first t elements of N_s, according to the optimal indexing defined above. Then ψ_{st} is defined by

$$\bar{N}_{st} = \bar{N}_s \cup N_s(t)$$

and

$$\delta_j^{st} = \begin{cases} \delta_j^s & \text{for all } j \in \bar{N}_s, \\ 0 & \text{for } j \in N_s(t), \\ 1 & \text{for } j = j_t. \end{cases} \qquad (j \neq j_t) \quad (8)$$

To show what this amounts to, we shall introduce the notation

$$\psi_s = (j_1, \cdots, \underline{j_h}, \cdots, j_k, \cdots, \underline{j_q}), \qquad (5a)$$

which is a shorthand for (5), and in which the nonunderlined indexes, like j_1, j_k, stand for $\delta^s_{j_1}=0$, $\delta^s_{j_k}=0$, while the underlined ones, like j_h, j_q, stand for $\delta^s_{j_h}=1$, $\delta_{j_q}=1$.

Now, if we were to use no choice rule and no tests, but simply start with the initial node, apply the branching procedure, then take each node thus generated, apply again the branching procedure, and so on, then a tree H would be generated that would be a partial graph of G, i.e., would contain all the nodes and part of the arcs of G. Figure 2 represents such a tree for $u=4$. In Fig. 2a we use the notation defined by (5a), while in 2b each solution ψ_s is written in the form of a sequence $\delta_{j_1}, \cdots, \delta_{j_q}$, the value of index j being given by the place of δ_j in the sequence.

In fact, however, this will not happen, and instead of H we shall construct the much 'smaller' tree T. Having selected a node s, i.e., a pseudo-solution ψ_s, we need not always generate *all* its descendants of the type (8). By constructing these descendants ψ_{st} in the increasing order of t (from left to right in Fig. 2), a simple test (*see* F-Test No. 2 below) applied to each new pseudo-solution ψ_{st} will show whether a feasible solution to F'_{st}, as defined by (6) in relation to ψ_{st}, may exist for any value of t larger than the current one. If not, then we can stop applying the branching procedure to the node s, and can select another node, i.e., start a new iteration.

On the other hand, each pseudo-solution ψ_{st} generated by the branching procedure will be checked (*see* F-Test No. 1) as to whether the solution associated with it by (7) is or is not feasible for F. If it is feasible, then the associated value z_{st} of the objective function will be marked.

So far we have supposed that the pseudo-solution ψ_s selected by our choice rule was such that the solution x^s associated with ψ_s by (7) was not feasible for F, i.e., z_s was not marked. Now, if z_s is marked, i.e., if x^s is a feasible solution to F, then it has 'passed through the filter,' i.e., it satisfies the (rather restrictive) necessary condition for feasibility in relation to P, which our filter-problem imposes. In this case x^s will be submitted to some tests derived from the additive algorithm (*see* the P-Test Step).

First, we shall check whether x^s is feasible for P (P-Test No. 1). If it is feasible, it will be shown to be optimal, and thus the procedure terminates. If it is not feasible, then we shall check (P-Test No. 2) whether a pseudo-solution ψ_v can exist, such that $\bar{N}_s \subset \bar{N}_v$, $\delta_j{}^v=\delta_j{}^s$, $\nexists j \epsilon \bar{N}_s$, and that the solution x^v associated with ψ_v by (7) should be feasible for P. If such a pseudo-solution ψ_v cannot exist, then ψ_s can be abandoned (discarded), since none of its descendants in any solution-tree generated by our branching procedure can yield a feasible solution to P. So in this case we shall continue our search by selecting another node according to our choice rule, i.e., by starting a new iteration.

If, however, the existence of ψ_v with the above property cannot be ex-

cluded (i.e., if the test is passed), then we check (P-Test No. 3) whether any of the free components of x^s must necessarily take on the value 0 (or the value 1) in any feasible solution to P. If no such free components are found, then we apply to ψ_s the branching procedure described above. If there are such components, then we assign them the corresponding values (*see* the Skipping Step) and thus obtain a new pseudo-solution, say ψ_u. [In this case, of course, the arc (s, u) of T, joining the nodes associated with ψ_s and ψ_u, respectively, will not belong to the tree H shown in Fig. 2).] Then we select another node for our branching procedure, i.e., start a new iteration.

Now let us return to the choice rule. Whenever we have to choose a node, i.e., whenever we start a new iteration, say $s+1$, we have a tree T_s of pseudo-solutions generated before the current iteration. All pseudo-solutions that (α) have no descendants in T_s, and (β) have not yet been submitted to P-Tests, will be called *active*. Let Π denote the set of pseudo-solutions ψ_i active at the current Choice Step, and let Z be the set of values $z_i = c\bar{x}^i$ (marked or not), associated with the pseudo-solutions $\psi_i \epsilon \Pi$, i.e.,

$$\Pi = \{\psi_i | \psi_i \text{ active}\}. \qquad Z = \{z_i = z(\psi_i) | \psi_i \epsilon \Pi\}. \tag{9}$$

Then our choice rule is to select ψ_s such that z_s should be minimal over Z. Thus, as the values z_i are nondecreasing along any path in G, the sequence of pseudo-solutions ψ_i selected under this rule is such that the sequence of associated values z_i is nondecreasing. This guarantees that the first solution x^s associated by (7) with a pseudo-solution ψ_s selected under our choice rule, which is feasible for P, is also optimal.

On the other hand, if at a certain stage of our procedure we find that $Z = \phi$, i.e., there are no active pseudo-solutions, then all the nodes of the solution-tree have been either explicitly or implicitly examined, with the conclusion that no feasible solution to P exists.

Statement of the Algorithm

The algorithm can be followed on the flow-chart in Fig. 3.

Phase I

Solve P' by one of the existing methods (*see*, for example, reference 24), and formulate the filter-problem F, using an optimal indexing (as defined in the previous section).

REMARK. The vector $a = \bar{u}A$ and the scalar $b_0 = \bar{u}b$ can be obtained directly from the simplex tableau associated with the optimal solution to P'. If DANTIZIG's[24] procedure is used, then $\bar{u}A = \tilde{c} + c$, where $\tilde{c}$ it the correspondent of c in the last simplex tableau resulting from the application of the procedure.

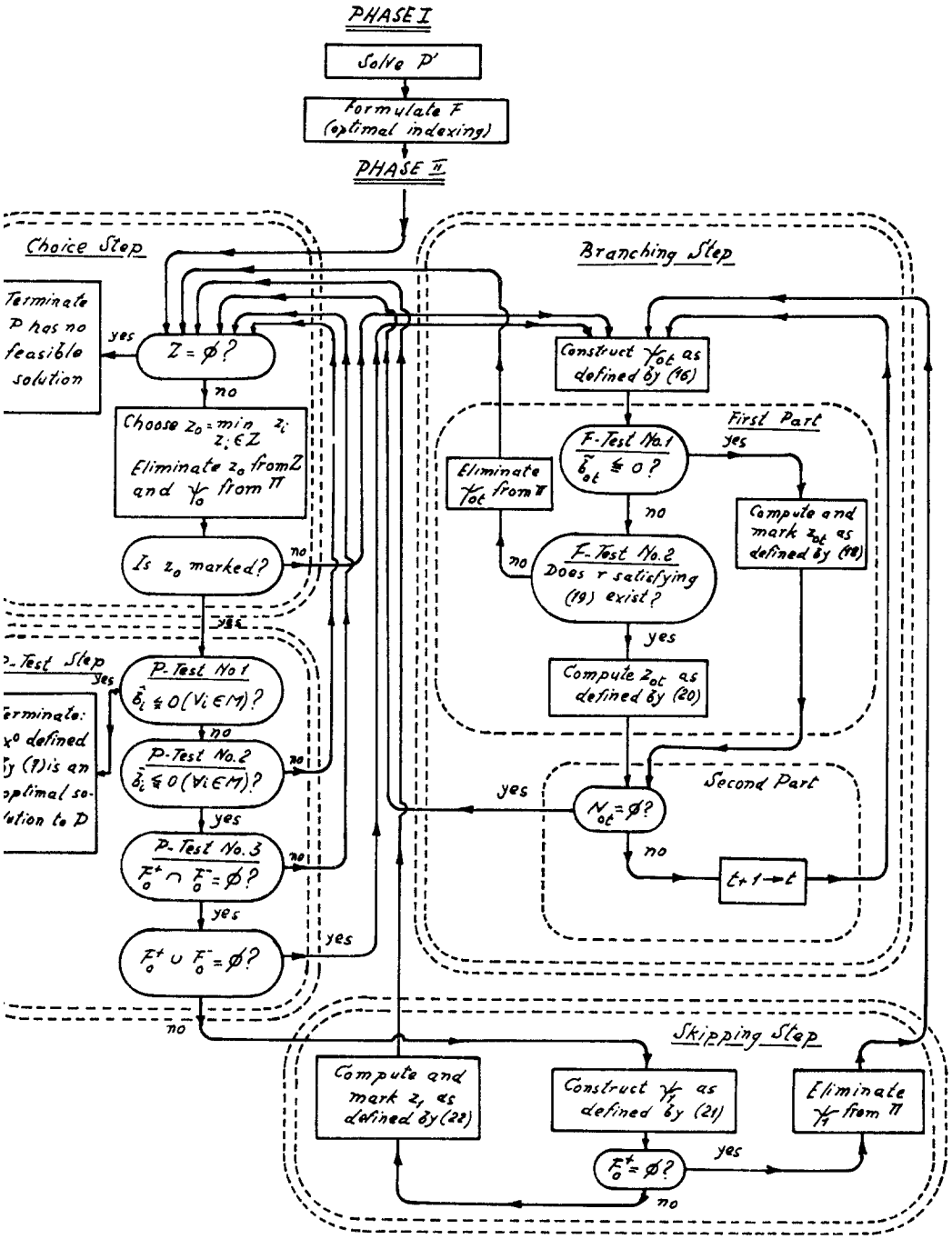

Fig. 3. Flow chart of the filter method

Phase II

At each iteration of this phase proceed as follows:

Choice Step

Examine the set Z. If $Z=\phi$, then P has no feasible solution and the algorithm terminates.

If $Z\neq\phi$, choose $z_0=\min_{z_i\in Z} z_i$, and eliminate z_0 from Z.

REMARK. At the start (first iteration), $Z=\{z_\alpha\equiv c\bar{x}\}$, where $\bar{x}$ is an optimal solution to P'.

If z_0 is not uniquely defined, choose (and eliminate from Z) one of those z_i satisfying the above definition for z_0. (Marked z_i should have priority over unmarked ones, if they satisfy the definition for z_0).

Let ψ_0 be the pseudo-solution associated with z_0, and let $\bar{N}_0$ be the set of indices it defines. Eliminate ψ_0 from Π.

Then, if z_0 is not marked, go to the Branching Step. If z_0 is marked go to the

P-TEST STEP

P-Test No. 1. Compute

$$\hat{b}_i=b_i-\sum_{j\in\bar{N}_0} a_{ij}\delta_j^0, \qquad (\nexists i\in M) \quad (10)$$

where δ_j^0 is the value of x_j in ψ_0, and $M=\{1,\cdots,m\}$.

If $\hat{b}_i\leqq 0$ $(\nexists i\in M)$, then x^0 defined by ψ_0 according to (7) is an optimal solution to P, and the algorithm terminates. If $\hat{b}_i>0$ for at least one $i\in M$, go to

P-Test No. 2. Compute

$$\bar{b}_i=\hat{b}_i-\sum_{j\in N_0^{i+}} a_{ij}, \qquad (\nexists i\in M^+) \quad (11)$$

where

$$N_0^{i+}=\{j\in N_0 | a_{ij}>0\}, \qquad M^+=\{i\in M|\hat{b}_i>0\}. \quad (12)$$

If $\bar{b}_i>0$ for at least one $i\in M^+$, go back to the Choice Step. If

$$\bar{b}_i\leqq 0 \quad (\nexists i\in M^+),$$

go to

*P-Test No. 3.** Form the sets

$$F_0^+=\bigcup_{i\in M^+}\{j\in N_0^{i+}|-a_{ij}<\bar{b}_i\}, \quad (13)$$

and

$$F_0^-=\bigcup_{i\in M^+}\{j\in N_0^{i-}|a_{ij}<\bar{b}_i\}, \quad (14)$$

where

$$N_0^{i-}=\{j\in N_0|a_{ij}<0\}. \quad (15)$$

* Generalized form of a test suggested by BERNHARD FLEISCHMANN.[27]

If $F_0^+ \cap F_0^- \neq \phi$, go back to the Choice Step.
If $F_0^+ \cup F_0^- = \phi$, go back to the Branching Step.
If $F_0^+ \cap F_0^- = \phi$, but $F_0^+ \cup F_0^- \neq \phi$, go to the Skipping Step.

BRANCHING STEP

Construct the pseudo-solutions ψ_{0t} $(t=1, 2, \cdots)$ such that

$$\bar{N}_{0t} = \bar{N}_0 \cup N_0(t)$$

and

$$\delta_j^{0t} = \begin{cases} \delta_j{}^0 & \text{for} \quad j \epsilon \bar{N}_0, \\ 0 & \text{for} \quad j \epsilon N_0(t),\ (j \neq k_t) \\ 1 & \text{for} \quad j = k_t, \end{cases} \tag{16}$$

where $N_0(t) = \{k_1, \cdots, k_t\}$ is the set formed by the first t elements of N_0 (according to an optimal indexing as defined in an earlier section), i.e., the set whose elements are the first t free indexes.

For each index t, proceed as follows:

FIRST PART. Apply

F-Test No. 1. Compute

$$\tilde{b}_{0t} = b_0 - \sum_{j \epsilon \bar{N}_{0t}} a_j \delta_j^{0t}. \tag{17}$$

If $\tilde{b}_{0t} \leqq 0$, compute and mark

$$z_{0t} = \sum_{j \epsilon \bar{N}_{0t}} c_j \delta_j^{0t}, \tag{18}$$

and go to the Second Part.

If $\tilde{b}_{0t} > 0$, apply

F-Test No. 2. Look for an index $r \epsilon N_{0t} = N - \bar{N}_{0t}$ such that

$$\sum_{\substack{j \epsilon N_{0t} \\ j \leqq r-1}} a_j < \tilde{b}_{0t} \leqq \sum_{\substack{j \epsilon N_{0t} \\ j \leqq r}} a_j. \tag{19}$$

If such an index r exists, then compute (but do not mark)

$$z_{0t} = \sum_{j \epsilon \bar{N}_{0t}} c_j \delta_j^{0t} + \sum_{\substack{j \epsilon N_{0t} \\ j \leqq r-1}} c_j + (c_r/a_r)\left(\tilde{b}_{0t} - \sum_{\substack{j \epsilon N_{0t} \\ j \leqq r-1}} a_j\right), \tag{20}$$

and go to the Second Part.

If no index r exists for which (19) is satisfied, eliminate ψ_{0t} from II and go back to the Choice Step.

SECOND PART.

If $N_{0t} = N - \bar{N}_{0t} \neq \phi$, augment by 1 the value of t, construct the next pseudo-solution defined by (16), and reapply the First Part of this Step. If $N_{0t} = \phi$, go back to the Choice Step.

SKIPPING STEP

Construct the pseudo-solution ψ_1 such that $\bar{N}_1=\bar{N}_0 \cup F_0^+ \cup F_0^-$ and

$$\delta_j^1=\begin{cases}\delta_j^0 & \text{for} \quad j\epsilon\bar{N}_0,\\ 1 & \text{for} \quad j\epsilon F^+,\\ 0 & \text{for} \quad j\epsilon F_0^-.\end{cases} \tag{21}$$

If $F_0^+\neq\phi$, compute and mark

$$z_1=\sum_{j\epsilon\bar{N}_0} c_j\delta_j^0+\sum_{j\epsilon F_0^+} c_j, \tag{22}$$

and go back to Choice Step.

If $F_0^+=\phi$, eliminate ψ_1 from II and go to the Branching Step. (In this case, N_1 takes the place of N_0 in the notation used with the Branching Step).

REMARK. If constraints of the type $\sum_j x_j\leqq 1$ are present, the procedure can be accelerated by the following modification of the Branching Step. Whenever a new pseudo-solution ψ_{st} is generated by assigning value 1 to x_{k_t} as defined in (16), check whether k_t belongs to any of the sets $\mathfrak{N}_{(i)}\subset N$ such that the ith constraint is $\sum_{j\epsilon\mathfrak{N}_{(i)}} x_j\leqq 1$. If this is the case for $i\epsilon\mathfrak{M}\subset M$, set $x_j=0$ for all $j\epsilon\{\cup_{i\epsilon\mathfrak{M}}\mathfrak{N}_{(i)}-\{k_t\}\}$, i.e., introduce all such j into $\bar{N}_{st}$.

THEOREM 2. *In a finite number of iterations, the accelerated additive algorithm yields either an optimal solution to P, or evidence that P has no feasible solution.*

Proof. Obviously, any feasible solution to P is also a feasible solution to F. To prove the convergence of the algorithm, we have to show that whenever the Choice Step selects a value $z_s\epsilon Z$, (or finds $Z=\phi$), this implies that no feasible solution x^r to P exists such that $z_r=cx^r<z_s$ (if $Z=\phi$, we can consider $z_s=\infty$); i.e., that all feasible solutions x^i to F, such that $z_i<z_s$, have been explicitly or implicitly examined and found not to be feasible for P. Then the first feasible solution to F obtained under the rules of the Choice Step and satisfying (2) is an optimal solution to P, and if no selection can be made because $Z=\phi$, then P has no feasible solution. As the number of solutions is finite, one of these two situations must occur in a finite number of iterations.

Now let us suppose that our Choice Step selects z_s and P-Test No. 1 shows that x^s defined by (7) in relation to ψ_s is a feasible solution to P. Further, let us suppose that x^s is not an optimal solution to P, i.e., there exists a feasible solution x^t to P such that $z_t=cx^t<z_s$. (An analogous reasoning holds for the case when $Z=\phi$ and we suppose that a feasible solution x^t to P still exists.)

Then consider the sequence of pseudo-solutions ψ_{i_1}, ψ_{i_2}, $\cdots$ obtained

from x^t by the following rule:

$$\delta'^{k}_{j}=\begin{cases} x_j{}^t & \text{for} \quad 1\leqq j\leqq j_k, \\ \text{undefined} & \text{for} \quad j>j_k, \end{cases} \qquad (k=1, 2, \cdots) \tag{23}$$

where

$$j_k=\max_{j<j_{k-1}}\{j|x_1{}^t=1\}, \tag{24}$$

with $j_0=n+1$. It is easy to see that the node i_1 associated with ψ_{i_1} in the tree H introduced in section 3 (Fig. 2) is the same as the node associated with x^t in the graph G (Fig. 1a), while the nodes associated with other pseudo-solutions of the sequence are the ancestors of node i_1 in H. Thus generating the sequence defined by (23) amounts to tracing the chain in H that connects node i_1 to the root of the tree H.

Now let T_s stand for the tree of pseudo-solutions generated until the current iteration, and let ψ_{i_k} be the first term of the sequence defined by (23), which corresponds to a node of T_s. Such a node obviously exists, since the root of H is the same as the root of T_s. The pseudo-solution ψ_{i_k} cannot be active, since $z_s=\min_Z z_i$ and the value of z is nondecreasing along all paths of H. Thus $\psi_{i_k}\epsilon\Pi$ would imply $z_s\leqq z_{i_k}\leqq z_t$, which contradicts the assumption that $z_t<z_s$.

This means that the pseudo-solution ψ_{i_k} has been submitted at a previous iteration to the P-Tests, with one of the following results:

(α) P-Test No. 2 has yielded $\bar{b}_i>0$ for some $i\epsilon M^+$, and ψ_{i_k} was therefore abandoned;

(β) P-Test No. 3 has yielded $F_0{}^+\cap F_0{}^-\neq\phi$ and ψ_{i_k} was therefore abandoned;

(γ) P-Test No. 3 has yielded $F_0{}^+\cup F_0{}^-=\phi$ and ψ_{i_k} was therefore submitted to the Branching Step;

(δ) P-Test No. 3 has yielded $F_0{}^+\cap F_0{}^-=\phi$ but $F_0{}^+\cup F_0{}^-\neq\phi$, and ψ_{i_k} was therefore submitted to the Skipping Step.

In case (α), obviously no solution associated with a descendant of ψ_{i_k} can be feasible for P. This also refers to x^t and thus contradicts our assumption.

In the other three cases, an examination of (13) and (14) shows that $F_0{}^+(F_0{}^-)$ is the set of those $j\epsilon N_0$ such that $x_j=1$ ($x_j=0$) in any feasible solution to P.

Thus, in case (β), P cannot have any feasible solution associated with a descendant of ψ_{i_k}.

In case (γ), the Branching Step applied to ψ_{i_k} would either have shown, through F-Test No. 2, that no descendant of ψ_{i_k} can yield a feasible solution to F' (and hence to P), or would have generated $\psi_{i_{k-1}}$, which would thus correspond to a node of T_s. But this contradicts our assumption that ψ_{i_k} is the first term of the sequence (23) corresponding to a node of T_s.

Finally, in case (δ) no descendant of ψ_{i_k} can yield a feasible solution to P, unless it is also a descendant of the pseudo-solution obtained from ψ_{i_k} by the Skipping Step, which is obviously not the case for x^t.

This proves theorem 2.

Discussion

The force of the method presented in the previous sections lies in the fact that

(α) the filter F obviously eliminates a large portion of the solution-graph G: the subset of nodes of G associated with solutions that are *not* feasible for F contains in most cases a very large part of the complete set of nodes; and (β) this elimination is obtained at a low cost: generating and testing solutions for F is easier than testing the same solutions for P.

The only serious difficulty that arises in connection with the implementation of this method, consists in the fact that all active pseudo-solutions $\psi_i \epsilon \Pi$ and the associated values $z_i \epsilon Z$ must be stored. We shall now shortly discuss one of the possible ways to overcome this difficulty.

Let us suppose that we have an (internal) storage capacity of Q pseudo-solutions ψ_i with their associated values z_i. Then we can proceed as follows:

(α) The pseudo-solutions ψ_i with the associated values z_i should be stored in the increasing order of z_i. This means that whenever a certain ψ_k is to be stored, it will be placed before the first ψ_i such that $z_i > z_k$. Then the Choice Step reduces to simply selecting the first $z_h \epsilon Z$.

(β) When the number of pseudo-solutions $\psi_i \epsilon \Pi$ attains Q, i.e., when the internal store gets filled up, then a certain part of Π, say the last $Q/2$ or $(Q+1)/2$ pseudo-solutions ψ_i and their associated values z_i, are to be transferred into the external memory, and this procedure is to be repeated as many times as the internal store gets filled up. Each time when a transfer to the external memory takes place, the last element of Π left in the inner store is to be labelled so as to guarantee that when it will be selected under the Choice Step, then the corresponding block from the external memory, or part of it, will be reintroduced in the working store. Whenever we reintroduce such a block into the working store, it has of course to be merged with the sequence already in the store, so that we get one single sequence of ψ_i rearranged in a nondecreasing order of the z_i.

Of course, the efficiency of such a procedure depends largely on the number of times the working store gets filled up, and it can be established only by practical testing, which has not yet been done. In any case, one is entitled to suppose that the size of problems manageable by the filter method will considerably exceed that of the problems solvable with the original additive algorithm.

In the next section we present two numerical examples. The second

one is a problem with twenty zero-one variables that has no special structure advantageous for the filter method, and which is solved by hand in 31 iterations, after the testing for P of 22 feasible solutions to F. The total number of solutions to P (feasible or not), which has thus been implicitly tested, is $2^{20}=1{,}048{,}576$.

Numerical Examples

EXAMPLE 1.

We shall consider example problem No. 1 of reference 3, written in the form of P introduced in the opening section of this paper:

$$\begin{aligned} 5x_1+7x_2+10x_3+3x_4+x_5&=\min,\\ x_1-3x_2+5x_3+x_4-4x_5&\geqq 2,\\ -2x_1+6x_2-3x_3-2x_4+2x_5&\geqq 0,\\ -x_2+2x_3-x_4-x_5&\geqq 1,\\ x_j=0 \quad \text{or} \quad 1 \ (j&=1,\cdots,5). \end{aligned}$$

Phase I

Solving the 'continuous' correspondent P' of P, we obtain the optimal solution

$$\tilde{x}=(0,\tfrac{1}{3},\tfrac{2}{3},0,0), \qquad \tilde{u}b=9,$$

and

$$\tilde{c}=\tilde{u}A-c=(-31\tfrac{1}{3},0,0,-52\tfrac{2}{3},-14\tfrac{1}{3});$$

hence

$$\tilde{u}A=\tilde{c}+c=(-16\tfrac{1}{3},7,10,-43\tfrac{2}{3},-11\tfrac{1}{3}).$$

We introduce the following optimal indexing

Original index	1	2	3	4	5
New index	4	1	2	5	3,

and formulate the filter-problem F in the reindexed variables x_j':

$$\begin{aligned} 7x_1'+10x_2'+x_3'+5x_4'+3x_5'&=\min,\\ 7x_1'+10x_2'-11\tfrac{1}{3}x_3'-16\tfrac{1}{3}x_4'-43\tfrac{2}{3}x_5'&\geqq 9.\\ [x_j'=0 \text{ or } 1 \ (j&=1,\cdots,5)] \end{aligned}$$

Phase II

Iteration 1.

CHOICE STEP. $Z=\{z_\alpha\}$. $z_0=z_\alpha=9$ is selected and eliminated from Z; ψ_α is eliminated from Π.

While using the notation introduced in earlier sections, or easier reference we shall also number the ψ_t and z_t in the order of their generation. Further, we shall use the shorthand notation $\psi_t=(j_1,\underline{j_2},\cdots,j_q)$, where j_h stands for $\delta^t_{j_h}=0$, and $\underline{j_k}$ for $\delta^t_{j_k}=1$.

BRANCHING STEP. $\bar{N}_0=\phi$, $\bar{N}_{01}=\{1\}$, $\psi_{01}=\psi_1=(\underline{1})$.

First Part

F-Test No. 1: $\tilde{b}_{01}=9-7=2>0$.

F-Test No. 2: We have $r=2$, $z_{01}=z_1=7+0+1(2-0)=9$.

Second Part. $N_{01}\neq\phi$, so we go to the

BRANCHING STEP. $\bar{N}_{02}=\{1,2\}$, $\psi_{02}=\psi_2=(1,\underline{2})$.

First Part.

F-Test No. 1: $\tilde{b}_{02}=9-0-10=-1\leqq 0$; $z_{02}=z_2|=10$.

Second Part. $N_{02}\neq\phi$.

BRANCHING STEP. $\bar{N}_{03}=\{1,2,3\}$, $\psi_{03}=\psi_3=(1,2,\underline{3})$.

First Part.

F-Test No. 1: $\tilde{b}_{03}=9-0-0+11\frac{1}{3}=38\frac{1}{3}>0$.

F-Test No. 2: There is no r satisfying (19). ψ_3 is eliminated from II.

Iteration 2.

CHOICE STEP. $Z=\{z_1,z_2{}^*\}$. $z_0=z_1=9$ is selected and eliminated from Z, while ψ_1 is eliminated from II.

BRANCHING STEP. $\bar{N}_0=\{1\}$, $\bar{N}_{01}=\{1,2\}$, $\psi_{01}=\psi_4=(\underline{1},\underline{2})$

First Part.

F-Test No. 1: $\tilde{b}_{01}=9-7-10=-8\leqq 0$; $z_{01}=z_4{}^*=17$.

Second Part. $N_{01}\neq\phi$.

BRANCHING STEP. $\bar{N}_{02}=\{1,2,3\}$, $\psi_{02}=\psi_5=(\underline{1},\underline{2},\underline{3})$.

First Part.

F-Test No. 1: $\tilde{b}_{01}=9-7+11\frac{1}{3}=17\frac{1}{3}>0$.

F-Test No. 2: There is no r satisfying (19). ψ_5 is eliminated from II, and z_4 from Z.

Iteration 3

CHOICE STEP. $Z=\{z_2{}^*,z_4{}^*\}$. $z_0=z_2{}^*=10$ is selected, ψ_2 and $z_2{}^*$ being eliminated from II and Z respectively. As $z_2{}^*$ is marked, we go to the

P-Test Step

P-Test No. 1: $\hat{b}_1=2-5=-3\leqq 0$,

$\hat{b}_2=0+3=3>0$,

$\hat{b}_3=1-2=-1\leqq 0$. As $\hat{b}_2>0$, we go to

P-Test No. 2: $\bar{b}_2=3-2=1>0$. As $\bar{b}_2>0$, we go to

Iteration 4

CHOICE STEP. $Z=\{z_4{}^*\}$. $z_0=z_4{}^*=17$, ψ_4 and $z_4{}^*$ are eliminated from II and Z respectively.

P-Test Step

P-Test No. 1: $\hat{b}_1=2+3-5=0$,

$$\hat{b}_2 = 0-6+3 = -3 \leqq 0,$$
$$\hat{b}_3 = 1+1-2 = 0.$$

As $\hat{b}_i \leqq 0$ $(i=1, 2, 3)$, the solution associated by (7) with ψ_4, namely

$$x_j' = \begin{cases} 1 & (j=1,2), \\ 0 & (j=3,4,5), \end{cases} \quad \text{i.e.,} \quad x_j = \begin{cases} 1 & (j=2,3), \\ 0 & (j=1,4,5), \end{cases}$$

is optimal.

EXAMPLE 2.

We shall now consider another pure zero-one problem with 20 variables and 3 constraints (constraints 2, 7, and 16 of test problem No. 16 of reference 14). The coefficient-matrix A and the vector c and b are given below. The problem will be denoted again by P.

$$A = \begin{Vmatrix} 6 & -5 & 8 & 3 & 0 & 1 & 3 & 8 & 9 & -3 & 8 & -6 & 3 & 8 & 6 & -7 & -6 & 2 & -3 & 7 \\ 1 & -3 & -3 & -4 & -1 & 0 & -4 & 1 & 6 & 0 & -8 & 0 & -1 & 5 & 4 & 1 & 9 & 7 & -2 & -2 \\ -3 & -6 & -1 & 3 & 5 & -6 & 9 & -6 & -3 & 9 & 6 & 3 & 6 & 6 & -6 & -2 & 7 & 6 & 0 & 7 \end{Vmatrix},$$

$$c = (3,\ 2,\ 5,\ 8,\ 6,\ 9,\ 11,\ 4,\ 5,\ 6,\ 11,\ 2,\ 8,\ 5,\ 8,\ 7,\ 3,\ 9,\ 2,\ 4),$$

$$b = \begin{Vmatrix} 21 \\ 10 \\ 14 \end{Vmatrix}.$$

Phase I

Solving the continuous correspondent of P yields the optimal solution

$$\bar{x}_9 = {}^{124}\!/_{141}, \quad \bar{x}_{10} = {}^{9}\!/_{47}, \quad \bar{x}_{17} = {}^{12}\!/_{47}, \quad \bar{x}_{14} = \bar{x}_{20} = 1,$$
$$\bar{x}_j = 0 \quad (j = 1, \cdots, 8,11,12,13,15,16,18,19);$$

with $\bar{u} = ({}^{73}\!/_{94},\ {}^{37}\!/_{282},\ {}^{87}\!/_{94})$ and $\bar{u}b = {}^{8623}\!/_{282} = 30.58$.

We have

$$\bar{u}A = (2.01;\ -9.83;\ 4.89;\ 4.58;\ 4.50;\ -4.78;\ 10.13;\ 0.80;\ 5;\ 6;\ 10.72;\ -1.88;\ 7.75;\ 12.42;\ -0.37;\ -7.16;\ 3;\ 8.02;\ -2.59;\ 11.65),$$

and we reindex the variables as follows (0 = original indexing; N = new indexing):

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
N	12	20	6	13	11	18	9	14	4	5	7	16	8	2	15	19	3	10	17	1.

Phase II

The problem is solved in 31 iterations. Table I contains all the relevant data of the process. The symbol i stands for the index of the pseudo-solution selected at the Choice Step of the current iteration, all pseudo-solutions being indexed in the order of their obtention, as in the previous example.

TABLE I

Iteration	i	z_i	ψ_i	Sequence of steps	Number of pseudo-solutions generated
1	α	15,5	—	B	9
2	1	15,5	1	B	10
3	10	15,5	1,2	B	11
4	20	15,5	1,2,3	B	11
5	21	15,5	1,2,3,4	B	10
6	22	15,5	1,2,3,4,5	B	10
7	23	15,6	1,2,3,4,5,6	B	8
8	25	17*	1,2,3,4,5,6,7,8	$P(1,2,3),S$	1
9	31	17*	1,2,3,4	$P(1,2,3),B$	12
10	33	17*	1,2,3,4,5,6	$P(1,2,3),B$	8
11	49	17*	1,2,3,4,5,6,7,8,9,10,11,12	$P(1,2)$	—
12	67	17*	1,2,3,4,5,6,7,8,9,10,11,12	$P(1,2)$	—
13	39	17,6	1,2,3,4,5,6,7,8,9,10,11,12	B	2
14	27	18*	1,2,3,4,5,6,7,8,9,10	$P(1,2,3),B$	7
15	32	18*	1,2,3,4,5	$P(1,2,3),B$	12
16	38	18*	1,2,3,4,5,6,7,8,9,10,11	$P(1,2,3),B$	5
17	58	18*	1,2,3,4,5,6,7,8,9,10,11,12	$P(1,2,3),S$	1
18	28	18	1,2,3,4,5,6,7,8,9,10,11	B	3
19	43	19*	1,2,3,4,5,6	$P(1,2,3),B$	12
20	60	19*	1,2,3,4,5,6,7,8,9,10,11,12,13,14	$P(1,2)$	—
21	29	19,9	1,2,3,4,5,6,7,8,9,10,11,12	B	2
22	24	20*	1,2,3,4,5,6,7	$P(1,2)$	—
23	27	20*	1,2,3,4,5,6,7,8,9	$P(1,2,3),S$	1
24	35	20*	1,2,3,4,5,6,7,8	$P(1,2,3),B$	10
25	40	20*	1,2,3,4,5,6,7,8,9,10,11,12,13	$P(1,2,3),S$	1
26	42	20*	1,2,3,4,5	$P(1,2,3),B$	13
27	48	20*	1,2,3,4,5,6,7,8,9,10,11	$P(1,2,3),B$	7
28	52	20*	1,2,3,4,5,6	$P(1,2,3),S$	1
29	66	20*	1,2,3,4,5,6,7,8,9,10,11	$P(1,2)$	—
30	78	20*	1,2,3,4,5,6,7,8,9,10,11,12	$P(1)$	—

The symbols used in the column containing the sequence of steps at each iteration, should read as follows: B=Branching Step; S=Skipping Step; P=P-Test Step. The numbers following P in parentheses refer to the P-Tests used. The Choice Step, which inaugurates each iteration, is not specified.

The total number of pseudo-solutions generated can be obtained by adding the numbers in the last column: 167. The maximum number of pseudo-solutions ψ_i (and of associated values z_i) to be kept in store during the procedure, which is equal to the size of Π (and of Z) at the last iteration, is $167-31=136$.

As we see, the pseudo-solution

$$\psi_{85}=(\underline{1},\underline{2},\underline{3},\underline{4},5,6,7,8,9,10,11,\underline{12})$$

selected at the 31st choice step, yields through (7) the (reindexed) optimal solution

$$x_j'=\begin{cases}1 & (j=1,2,3,4,12),\\ 0 & (j=5,\cdots,11,13,\cdots,20),\end{cases}$$

which corresponds to

$$x_j=\begin{cases}1 & (j=1,9,14,17,20),\\ 0 & (j=2,\cdots,8,10,\cdots,13,16,18,19)\end{cases}$$

in the original indexing.

This solution has been obtained after testing for P 32 solutions, out of a total of $2^{20}=1{,}048{,}576$.

EXTENSION TO THE MIXED-INTEGER CASE BY BENDERS' PARTITIONING PROCEDURE

The Mixed-Integer Zero-One Problem

In the previous section we have called the attention of the reader to the practical importance of linear programming with zero-one variables. We shall add now, that in most of the practical instances, the zero-one programming problem appears not in its pure form, but as a mixed-variables problem, in which part of the variables are continuous. The problem may be formulated as that of finding $(x, y) = (x_1, \cdots, x_n; y_1, \cdots, y_p)$ such that

$$c^1x+c^2y=\min, \tag{25}$$

$$A^1x+A^2y\geqq b, \tag{26}$$

$$x\geqq 0, \tag{27}$$

$$y_k=0 \text{ or } 1, \qquad (k=1,\cdots,p) \tag{28}$$

where $c^1=(c_j{}^1)$, $c^2=(c_k{}^2)$, $A^1=(a_{ij}^1)$, $A^2=(a_{ik}^2)$, and $b=(b_i)$, $(i=1,\cdots,m;$ $j=1,\cdots,n;\ k=1,\cdots,p)$ are given. Let this problem be $\mathcal{P}$.

J. F. BENDERS[25] has given a partitioning procedure for solving mixed-variables (partly linear, partly nonlinear) mathematical programming problems, in which the nonlinear variables belong to an arbitrary set S.

This procedure requires the alternate solution of two subproblems, one of which is a linear program, while the second one is a mathematical programming problem whose properties depend on S. If this procedure is applied to our case, the second subproblem takes on the form of a linear program with p zero-one variables and one arbitrary (i.e., not necessarily nonnegative) continuous variable.

For a detailed description of Benders' procedure the reader is referred to reference 25 (*see also* reference 13 for an alternative presentation and a very useful discussion of the method). Here we shall confine ourselves to a brief restatement of the procedure (as applied to our special case) and shall then concentrate on showing how the filter method can be used to advantage for the solution of the nonlinear subproblem.

Introducing an unrestricted scalar variable y_0, condition (25) can be replaced by

$$y_0 = \min, \tag{25a}$$

$$y_0 \geqq c^1x + c^2y, \tag{25b}$$

and then the two subproblems $\mathcal{P}^1$ and $\mathcal{P}^2$ derived from $\mathcal{P}$ can be written as

$\mathcal{P}^1$

$$y_0 = \min, \tag{29}$$

$$y_0 - c^2y + u^rA^2y \geqq u^rb, \qquad (r=1, \cdots, t) \tag{30}$$

$$u^sA^2y \geqq u^sb, \qquad (s=1, \cdots, w) \tag{31}$$

$$0 \leqq y_k \leqq 1, \qquad (k=1, \cdots, p) \tag{32}$$

$$y_k \text{ integer}, \tag{33}$$

$$y_0 \text{ arbitrary}. \tag{34}$$

$\mathcal{P}^2$

$$c^1x = \min, \tag{35}$$

$$A^1x \geqq b - A^2\tilde{y}, \tag{36}$$

$$x_j \geqq 0, \qquad (j=1, \cdots, n) \tag{37}$$

where $u^r = (u_i^r)$, $u^s = (u_i^s)$, $i=1, \cdots, m$, and $\tilde{y} = (\tilde{y}_k)$, $k=1, \cdots, p$, are defined as follows. The vectors u^r in $\mathcal{P}^1$ are extreme points of the set $\{u \mid uA^1 \leqq c^1, u \geqq 0\}$, i.e., basic solutions to the dual of $\mathcal{P}^2$ (optimal for certain values of $\tilde{y}$), while u^s are direction vectors of the extreme rays of the cone $\{u \mid uA^1 \leqq 0, u \geqq 0\}$. On the other hand, the vector $\tilde{y}$ in $\mathcal{P}^2$ is the y-component of an optimal solution $(\tilde{y}_0, \tilde{y})$ to $\mathcal{P}^1$.

In the process of alternately solving $\mathcal{P}^1$ and $\mathcal{P}^2$, each time we solve $\mathcal{P}^1$, either the procedure comes to an end, or we obtain a new vector $\tilde{y}$ for $\mathcal{P}^2$;

and each time we solve $\mathcal{P}^2$, either the procedure ends, or we obtain a new constraint (30), or a new constraint (31), or both, for $\mathcal{P}^1$.

More precisely, the two stages of the procedure run as follows:

STAGE *A*. We solve $\mathcal{P}^1$

A1. If $\mathcal{P}^1$ is infeasible, $\mathcal{P}$ has no feasible solution.

A2. If $\mathcal{P}^1$ has an optimal solution $(\tilde{y}_0, \tilde{y})$, or if it has no finite optimum but $(y_0, \tilde{y})$ is a feasible solution for whatever small values assigned to y_0, then $\tilde{y}$ defines a new right-hand side for the constraint set of $\mathcal{P}^2$, and we go to stage *B*.

STAGE *B*. We solve $\mathcal{P}^2$

B1. If $\mathcal{P}^2$ has no finite optimum, then $\mathcal{P}$ has no finite optimum.

B2. If $\mathcal{P}^2$ has an optimal solution $\tilde{x}$ such that $c^1\tilde{x} \leqq \tilde{y}_0 - c^2\tilde{y}$, then $(\tilde{x}, \tilde{y})$ is an optimal solution to $\mathcal{P}$. If, however, $c^1\tilde{x} > \tilde{y}_0 - c^2\tilde{y}$, then an optimal solution $\tilde{u}$ to the dual of $\mathcal{P}^2$ defines a new constraint of the form (30) for $\mathcal{P}^1$, and we go back to stage *A*.

B3. If $\mathcal{P}^2$ has no feasible solution, and the objective function of its dual tends to infinity along a halfline $\{v | v = \tilde{u} + \lambda\bar{u}\}$, where $\tilde{u}$ is a vertex of $\{u | uA^1 \leqq c^1, u \geqq 0\}$ and $\bar{u}$ a direction-vector of an extreme ray of $\{u | uA^1 \leqq 0, u \geqq 0\}$, then $\bar{u}$ always defines a new constraint of the form (31) for $\mathcal{P}^1$; and if $\tilde{u}(b - A^2\tilde{y}) > \tilde{y}_0 - c^2\tilde{y}$, then $\tilde{u}$ also defines a constraint of the form (30) for $\mathcal{P}^1$.

Thus we go back to stage *A*.

Concentrating now upon the way of solving $\mathcal{P}^1$, we see that this subproblem has some special features: (a) besides the 0-1 variables y_k $(k = 1, \cdots, p)$, it contains the unrestricted variable y_0; (b) y_0 has positive or zero coefficients in all constraints of the form $\geqq$; (c) the objective function does not contain any of the 0-1 variables y_k; and (d) we have to solve repeatedly the same problem with one or two constraints being added each time.

We shall now describe a specialized version of the filter method for the above problem. First we solve the continuous correspondent of $\mathcal{P}^1$, i.e., the problem obtained from $\mathcal{P}^1$ by dropping (33). Of course, to do this we have to substitute $y_0 = y_0' - y_0''$ and replace (34) through (34′) $y_0', y_0'' \geqq 0$. On the other hand, as the current problem $\mathcal{P}^1$ differs from the problem $\mathcal{P}^1$ solved in the previous iteration only by one or two new constraints, solving the continuous correspondent of the current $\mathcal{P}^1$ reduces in fact to post-optimizing the solution obtained for the continuous correspondent of the previous problem $\mathcal{P}^1$.

Then we formulate (*see* subsection, "The Filter") the filter problem *F*

$$y_0 = \min, \tag{38}$$

$$a_0 y_0 + ay \geqq b_0, \tag{39}$$

$$0 \leqq y_k \leqq 1, \qquad (k = 1, \cdots, p) \tag{40}$$

$$y_k \text{ integer}, \tag{41}$$

$$y_0 \text{ arbitrary}, \tag{42}$$

where (a_0, a) and b_0 correspond respectively to a and b_0 in $(2')$.

From property (b) above of $\mathcal{P}^1$ and from duality theory it follows that $a_0=0$ only if y_0 has zero coefficients in all constraints, i.e., there are no constraints of type (30). But in this case $\mathcal{P}^1$ obviously has no finite optimum, y_0 can be set $-\infty$, and solving $\mathcal{P}^1$ reduces to finding a feasible y-vector, i.e., finding a feasible solution to a pure 0-1 problem that can easily be done for instance with the additive algorithm as described in reference 3. So we shall leave aside this case and shall suppose that $a_0 \neq 0$.

Further, we shall suppose that $a_k \leqq 0, \nsucc k$. (Whenever this is not the case for a certain k, one can substitute $y_k' = 1 - y_k$.) This assumption guarantees that the solution $\tilde{y}_k = 0, \nsucc k$, $\tilde{y}_0 = b_0/a_0$ is feasible and optimal for F. Also, we suppose that an optimal indexing as defined in the subsection "Generating and Testing Solutions" has been introduced, i.e., $a_{k_1} > a_{k_2} \Rightarrow k_1 < k_2$. It should be mentioned that this indexing has to cover all indices $k = 1, \cdots, p$, also those for which $a_k = 0$, so that the latter should not be omitted in the process of generating pseudo-solutions.

We then apply phase II of the filter method with the following modifications.

The branching procedure described in the subsection "Generating and Testing Solutions" will be used as before to generate feasible solutions to F in such an order that (α) the sequence of associated values y_0 is non-decreasing, and (β) when a certain value y_0^s has been reached (i.e., selected for branching) then all feasible solutions (y_0, y) to F such that $y_0 < y_0^s$ have been (explicitly or implicitly) enumerated. But, as y_0 is unrestricted, a 0-1 vector y^s associated with a pseudo-solution ψ_s by a relation of type (7) (we are using the notations of the subsection cited) always defines a feasible solution (y_0^s, y^s) to F through

$$y_0^s = (b_0 - a y^s)/a_0^i, \tag{43}$$

which, for the given y^s, yields the 'optimal' y_0^s. Thus (43) replaces (18) and F-Tests no. 1 and 2 can be dropped. Also, there is no need to mark any value y_0^s associated with a pseudo-solution ψ_s, because all values y_0 found under the procedure correspond to feasible solutions to F. Thus the First Part of the Branching Step in our algorithm reduces to the computing of y_0^s according to (43). Then we go to the Second Part which remains unchanged.

Accordingly, the Choice Step of our algorithm will in this version always have to be followed by the P-Test Step. In other words, whenever we choose

$$y_0^s = \min_z y_0^z \tag{44}$$

the 0-1 vector y^s associated with y_0^s will have to be submitted to the P-Tests.

Now P-Test No. 1, which is a feasibility-test with regard to $\mathcal{P}^1$ for the vector y^s associated with the pseudo-solution ψ_s selected under the Choice Step, will have to be modified as follows.

Let $M_1=\{1, \cdots, t\}$ be the set of row indices in (30), and $M_2=\{1, \cdots, w\}$ the set of row indices in (31). Given a 0-1 vector y^s satisfying (31), in view of property (b) of $\mathcal{P}^1$ a scalar ζ^s can always be found, such that (y_0, y^s) is a feasible solution to $\mathcal{P}^1$ for any $y_0 \geqq \zeta^s$, but is not feasible for any $y_0 < \zeta^s$. Moreover, this scalar is easily seen to be

$$\zeta^s = \hat{b}_{i_0} = \max_{i \epsilon M_1} \hat{b}_i, \tag{45}$$

where the $\hat{b}_i$ are defined by (10).

Then our modified test will consist of two parts—let them be P-Test No. 1a and No. 1b. First, under P-Test No. 1a, we compute $\hat{b}_i$ for all $i \epsilon M_2$. If $\hat{b}_i > 0$ for some $i \epsilon M_2$, then the vector y^s associated with ψ_s by (7) does not yield a feasible solution to $\mathcal{P}^1$, and we go to P-Test No. 2.

If, however, $b_i \leqq 0$ for all $i \epsilon M_2$, or if $M_2 = \phi$, then y^s obviously yields a feasible solution to $\mathcal{P}^1$, and the y_0-component of the 'best' solution to $\mathcal{P}^1$ associated with y^s is ζ^s as defined by (45). Thus we compute $\hat{b}_i$ for all $i \epsilon M_1$ and ζ^s.

Our procedure requires that we always keep track of the smallest value ζ so far obtained. Thus, if ζ_*^{s-1} stands for the smallest value ζ obtained before the current iteration, then after computing ζ^s (the new value obtained for ζ in the current iteration) we choose

$$\zeta_*{}^s = \min[\zeta_*^{s-1}, \zeta^s], \tag{46}$$

we record $\zeta_*{}^s$ and go to P-Test 1b.

This simply consists of checking whether there are any free indices $k \epsilon N_s$ left, such that $a_{i_0 k} > 0$, where i_0 is defined by (45). If there are not, then obviously no descendant of the node corresponding to our current solution in the tree H can yield a better solution than the current one, and so we do not branch but go back to the Choice Step. If, however, there is at least one index $k \epsilon N_s$ with the above property, then we go to the Branching Step.

As to P-Tests No. 2 and 3, they are to be applied in the same way and with the same conclusions as in subsection "Statement of the Algorithm," but only to the constraints $i \epsilon M_2$, which means that one has to write $i \epsilon M_2$ instead of $i \epsilon M$ in (11), (13), and (14).

In case P-Test No. 3 is not passed and one is led to the Skipping Step, this is also to be applied in the same way as in the original version of the method with the only difference that (22) is replaced by (43).

Coming now again to our Choice Step, each time we select an element

$y_0^s \in Z$, we shall compare it with ζ_*^{s-1}. If $y_0^s < \zeta_*^{s-1}$, the procedure can go on, i.e., we go to the P-Test Step. Otherwise the algorithm terminates, because we have

THEOREM 2. *If $y_0^s \geqq \zeta_*^{s-1}$, the solution associated with ζ_*^{s-1} is optimal for $\mathcal{P}^1$.*

Proof. Any feasible solution to $\mathcal{P}^1$ is a feasible solution to F. So if

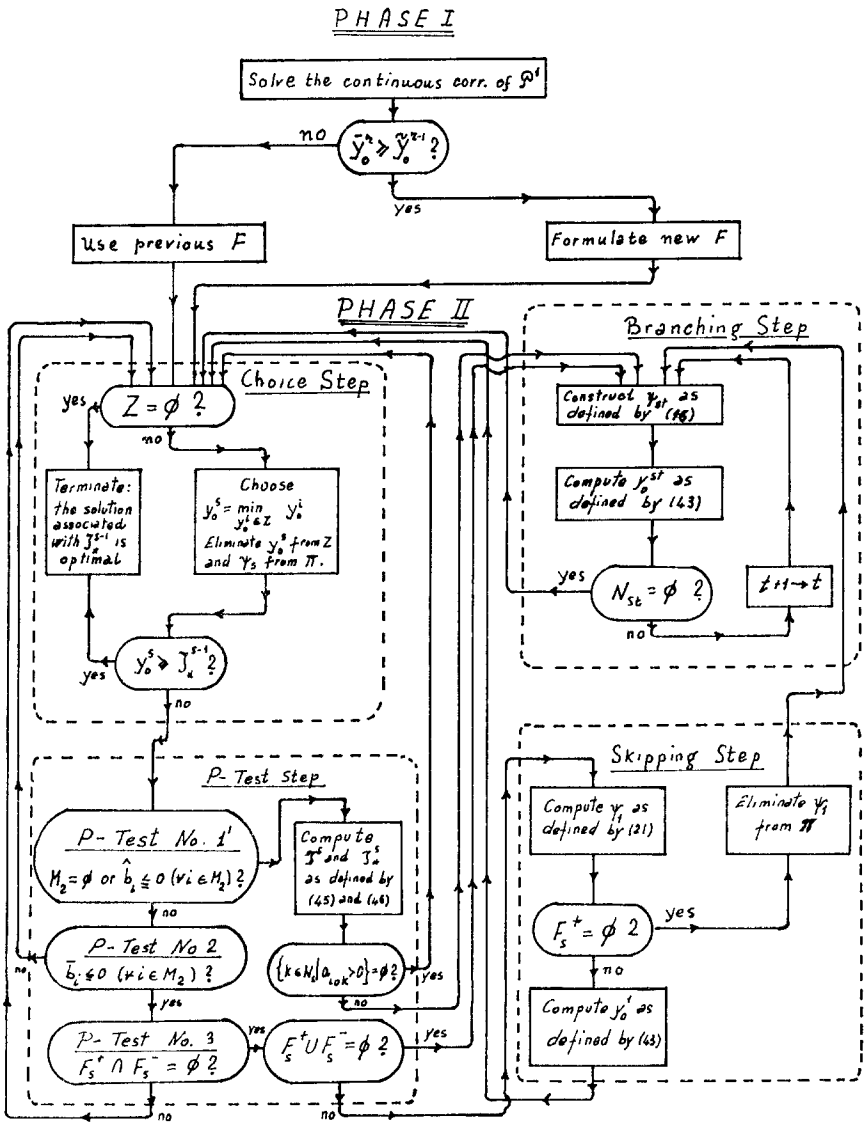

Fig. 4. Flow chart of the modified filter method for solving $\mathcal{P}^1$.

there exists a feasible solution $(\hat{\xi}, \hat{y})$ to $\mathcal{P}^1$ such that $\hat{\xi} < \zeta_*^{s-1}$, then it is also feasible for F. But this means that a feasible solution $(\hat{\xi}, \hat{y})$ to F exists such that $\hat{\xi} < y_0^s$, which has not been so far generated under our branching procedure. However, reasoning along the lines of the proof of theorem 1 we find that whenever a solution (y_0^s, y^s) is selected for branching under the rules of our procedure, all solutions (y_0, y) to F such that $y_0 < y_0^s$ had been explicitly or implicitly enumerated. So our assumption leads to a contradiction.

Theorem 2 shows one situation in which the method finds an optimal solution to $\mathcal{P}^1$. But what happens if that situation does not occur, i.e., if we never have $y_0^s \geqq \zeta_*^{s-1}$. This is possible in principle, though it is not likely to occur in practice. In any case, if this happens, i.e., if $Z = \phi$ at the Choice Step of the sth iteration, then the solution associated with ζ_*^{s-1} is optimal. Indeed, $Z = \phi$ means that all solutions to $\mathcal{P}^1$ have been explicitly or implicitly enumerated—and then obviously the one with the lowest-valued objective function is the optimal solution.

Finally, we shall also introduce a modification meant to take advantage of property (d) of $\mathcal{P}^1$.

Let us suppose we have completed a number of iterations in the process of solving $\mathcal{P}$, i.e., we have a number of times alternately solved $\mathcal{P}^1$ and $\mathcal{P}^2$, and let $\mathcal{P}_r^1$ stand for our current problem $\mathcal{P}^1$. If $\bar{y}_0^r$ denotes the y_0-component of an optimal solution to the continuous correspondent of $\mathcal{P}_r^1$, and $\tilde{y}_0^{r-1}$ stands for the y_0-component of the optimal solution to $\mathcal{P}_{r-1}^1$, i.e., of the optimal solution obtained for $\mathcal{P}^1$ (not for its continuous correspondent!) at the previous iteration, then the adoption of the following rule is recommended:

(α) If $\bar{y}_0^r \geqq \tilde{y}_0^{r-1}$, apply the filter method in the way shown above (i.e., formulate a new filter problem, etc.)

(β) If, however, $\bar{y}_0^r < \tilde{y}_0^{r-1}$, then use the filter problem of the previous iteration. In this case continue the generation of pseudo-solutions from the point where it had been stopped (i.e., start with the sets II and Z and the value ζ_* obtained at the end of the previous iteration).

This modified version of the filter method for solving $\mathcal{P}^1$ is summarized on the flow-chart shown in Fig. 4.

As we see, in this special version of the method the role of our F-problem is not so much that of a 'filter' as that of a 'guide' to our search: by the order in which it generates solutions to be tested for $\mathcal{P}^1$, it enables us to eliminate from our search all solutions that would be enumerated after reaching (y_0^s, y^s) such that $y_0^s \geqq \zeta_*^{s-1}$.

The General Integer and the Mixed-Integer Problem

We shall now consider the problem

$$c^1x^1 + c^2x^2 = \min, \tag{47}$$

$$A^1x^1+A^2x^2 \geqq b, \tag{48}$$

$$x_j^1,\ x_k^2 \geqq 0, \qquad (j=1,\cdots,n;\ k=1,\cdots,p) \tag{49}$$

$$x_k^2 \text{ integer} \leqq L_k. \qquad (k=1,\cdots,p) \tag{50}$$

This is a linear program in which some of the variables, having finite upper bounds L_k, are constrained to take on integer values. Obviously, the pure integer programming problem is a special case of the above one, in which $\{1,\cdots,n\}=N=\phi$.

Any problem of the above type can be reduced to a mixed-integer (a pure integer, if $N=\phi$) linear program with zero-one variables, by substituting for each x_k^2 its binary representation (*see* reference 4). However, this procedure implies an increase of the number of variables by several times. We shall show now that any problem of the above type can be solved with the method described in the previous section, without using the binary representation of the integer variables.

For any fractional number α, let $[\alpha]$ stand for the largest integer inferior to α, and $\langle\alpha\rangle$ for the smallest integer superior to α. We then proceed as follows:

1. Solve the continuous correspondent of (47), (48), (49), and (50). If this problem has no feasible solution, then neither has the original one. If it has a feasible solution but no finite optimum, then the original problem has either no feasible solution, or no finite optimum. Let us suppose that the continuous problem has a (finite) optimal solution $(\bar{x}^1,\ \bar{x}^2)$. If $\bar{x}_k^2$ is integer for $k=1,\cdots,p$, then we have found an optimal solution to our original problem. If this is not the case, denote $D=\{k|\bar{x}_k^2 \text{ is fractional}\}$ and go to 2.

2. Apply the method of the previous section to the problem $\mathcal{P}$:

$$y_0=\min, \tag{47a}$$

$$y_0-c^1x^1-c^2x^2 \geqq 0, \tag{47b}$$

$$A^1x^1+A^2x^2 \geqq b, \tag{48}$$

$$x_j^1,\ x_k^2 \geqq 0, \tag{49}$$

$$\left.\begin{array}{l} x_k^2+\langle\bar{x}_k^2\rangle y_k \geqq \langle\bar{x}_k^2\rangle, \\ -x_k^2-L_k y_k \geqq -[\bar{x}_k^2]-L_k, \\ y_k=0 \text{ or } 1. \end{array}\right\} \qquad (k\epsilon D) \tag{50'}$$

Here the constraints (50′) translate the disjunctions

$$(x_k^2 \geqq \langle\bar{x}_k^2\rangle) \vee (x_k^2 \leqq [\bar{x}_k^2]). \tag{50''}$$

In the course of computation, Stage B (solving $\mathcal{P}^2$) has two possible

outcomes: (a) all x_2^k are integers, (b) some x_2^k are fractional. In case (a), we continue as usual, turning to $\mathcal{P}^1$. In case (b), we introduce an additional pair of constraints (50′) for each noninteger x_2^k, and again solve $\mathcal{P}^2$. We repeat this until situation (a) prevails. Of course, each newly introduced 0-1 variable gets zero coefficients in all previous constraints of $\mathcal{P}^1$.

THEOREM 3. *The procedure 1, 2 ends in a finite number of iterations.*

Proof. The set $\{1, \cdots, p\}$ is finite and $x_k^2 \leqq L_k$ $(k=1, \cdots, p)$. Thus the number of pairs of constraints (50″) one might have to introduce is finite, and always less than $\sum_{k=1}^{k=p} L_k$.

This may not sound encouraging, but a theoretical proof should not be taken for a statement about efficiency. There are reasons to believe that the number of 0-1 variables one has to use need not substantially exceed the number of integer variables.

Given the special structure of $\mathcal{P}$, solving the current subproblem $\mathcal{P}_s^2$ at each stage B reduces to post-optimization. Only one of each pair of constraints (50′) needs to be considered (namely that which is 'effective,' i.e., the first one if $\tilde{y}_k=0$ and the second one if $\tilde{y}_k=1$). Moreover, these constraints need not be explicitly introduced into $\mathcal{P}^2$. Instead, for each constraint of the type $x_h^2 \geqq \langle \bar{x}_h^2 \rangle$, substitute

$$x_h^2 = \langle \bar{x}_h^2 \rangle + x_h^3, \qquad x_h^3 \geqq 0, \tag{51}$$

and for each constraint of the type $x_k^2 \leqq [\bar{x}_k^2]$ substitute

$$x_k^2 = [\bar{x}_k^2] - x_k^3, \qquad x_k^3 \geqq 0. \tag{52}$$

(Observe that the coefficient of x_h^3 in the objective function will be c_h^2, while that of x_k^3 will be $-c_k^2$).

Thus the fractional variables x_h^2, x_k^2 are replaced through (51) or (52) by nonnegative variables x_h^3, x_k^3, which take on *negative* values in the current solution $(x_h^3 = \bar{x}_h^2 - \langle \bar{x}_h^2 \rangle < 0$, $x_k^3 = [\bar{x}_k^2] - \bar{x}_k^2 < 0)$, and therefore must be eliminated from the basis. This can be done by one or more dual simplex iterations. The feasible (optimal) solution $(\bar{x}^1, \bar{x}^2)$ that one obtains for $\mathcal{P}_s^2$ is obviously the same as if the constraints (50′) had been explicitly introduced.

On the other hand, for defining a new constraint to $\mathcal{P}_{s+1}^1$ we need the optimal solution $\tilde{u}$ of the dual of $\mathcal{P}_s^2$, associated with $(\bar{x}^1, \bar{x}^2)$. How can we obtain the components $\tilde{u}_i$ associated with the constraints (50′) that were not explicitly introduced into $\mathcal{P}_s^2$? Well, by the procedure described above, a new variable x_j^3 is introduced for each neglected constraint of the type (50′). The dual variable associated with the nonnegativity constraint on x_j^3 (whose value appears in the last row of the optimal simplex tableau, Table I, in the column corresponding to x_j^3) takes the place

(and takes on the value) of the dual variable associated with the neglected constraint.*

3. APPLICATION TO MACHINE-SEQUENCING BY DISJUNCTIVE GRAPHS

The Model

The model we shall discuss in this section (for earlier presentations *see* references 36 and 37) translates a great variety of scheduling problems. Best known among them is the machine-sequencing problem (*see* references 28–35).

Suppose that we have m items (or lots of items) to be processed on q machines. Each item i is to be submitted to p operations ($i=1, \cdots, m$). Each operation i is to be carried out on a certain machine. The order in which the various operations concerning one item are to be carried out is fixed by the technological process. On the other hand, the order in which the various items are to be processed by a certain machine is not fixed. The problem is to find such an order (sequence) of carrying out the various operations on the various machines that the total time spent for processing all items should be minimal.

Let $M=\{1, \cdots, m\}$, $Q=\{1, \cdots, q\}$, and let us reindex the operations so that any operation to be applied to a certain item should have an index different from the index of the same operation to be applied to another item. This will be done by setting $p_i=n_i-n_{i-1}$ for $i=1, \cdots, m$, with $n_0=0$. Let $N_i=\{n_{i-1}+1, \cdots, n_i\}$ and $N=\bigcup_{i=1}^{m} N_i=\{1, \cdots, n_m\}$. Then any index $j \epsilon N$ uniquely defines a certain operation to be applied to a certain item on a certain machine. Let t_j denote the time (moment) of starting operation j, let d_j stand for the time (interval) necessary for carrying out operation j, and let N^k be the set of indices j associated with the operations to be executed on machine k. Obviously $d_j>0$ for all $j \epsilon N$. Further, $\bigcup_{k=1}^{q} N^k=N$ and $N^h \cap N^k=\phi$ for any $h \epsilon Q$, $k \epsilon Q$, $h \neq k$. Let $t_0=0$ and let t_n be the time (moment) of ending the processing of all items. Then our problem can be written in the following form:

Find t_n, t_j ($j=1, \cdots, n_m$) satisfying

$$t_n = \min, \tag{53}$$

$$\begin{aligned} t_{j+1}-t_j &\geqq d_j, \qquad (j \epsilon N_i-\{n_i\},\ i \epsilon M), \\ t_n-t_{n_i} &\geqq d_{n_i}, \qquad (i \epsilon M), \end{aligned} \tag{54}$$

* The approach outlined in this section has directed the author's attention toward certain duality relations in mixed-integer programs. Further research has led to the formulation of a pair of dual mixed-integer programming problems, which have the property of involution and to which most of the results of linear programming duality theory have been extended.[46] This, however, forms the object of another paper.

$$t_n,\ t_j \geqq 0, \qquad (j \epsilon N), \tag{55}$$

$$(t_j - t_h \geqq d_h) \vee (t_h - t_j \geqq d_j), \quad (h \epsilon N^k,\ j \epsilon N^k,\ k \epsilon Q). \tag{56}$$

Let us denote $\mathcal{P}$ as the problem (53), (54), (55), and (56).

In the above model (which is identical with the one used in references 34 and 35) the constraints (54) express the order fixed by the technological process for the operations to be applied to each item, while the disjunctions (56) translate the requirement that there should be no overlapping between the time intervals assigned to the execution of any two operations that use the same machine. Although the condition prescribed by (56) for any pair of indices h,j belonging to one and the same set N^k is that *at least* one of two inequalities should hold, the fact that $d_j > 0$ for all $j \epsilon N$ also implies that *only one* of the inequalities can hold. It is easy to see that $\mathcal{P}$ always has a finite optimal solution.

We chose for our model the typical case when order relations are given only between operations related to the same item. However, in practice we often have to face situations when order relations also occur between operations related to different items. For instance, if a certain item k results from the assembly of items i and j, then obviously the first operation to be carried out on item k will have to start after the last operations on each of the two separate items i and j have been ended. But it is easy to see that the relations

$$\begin{aligned} t_{n_{k-1}+1} - t_{n_i} &\geqq d_{n_i}, \\ t_{n_{k-1}+1} - t_{n_j} &\geqq d_{n_j}, \end{aligned} \tag{54'}$$

expressing this condition are of exactly the same type as (54) and thus their introduction makes no difference to our model.

Graph-Theoretical Formulation

If we exclude from P the disjunctions (56), we obtain a linear program of a special type, known as a critical path- or PERT-problem. The graph (PERT network) G that can be associated with this problem is shown in Fig. 5a.

Each node j of G represents the event of starting an operation j ($j = 1, \cdots, n_m$), and corresponds to a variable t_j. Node 0 (the source) represents the event: 'any one of the operations $n_{i-1}+1$, $i = 1, \cdots, m$, can be started'; while node n (the sink) marks the event: 'all operations finished.' An arc $(j, j+1)$ is associated with each operation j, such that $j \epsilon N_i - \{n_i\}$, $i \epsilon M$ and with each constraint of the type $t_{j+1} - t_j \geqq d_j$; while each operation n_i, $i \epsilon M$ and each constraint of the type $t_n - t_{n_i} \geqq d_{n_i}$, $i \epsilon M$, is represented by an arc (n_i, n). For any $j \epsilon N$, the time interval d_j is the length of the arc associated with operation j. A path H in G is defined as usually,[26] and

G

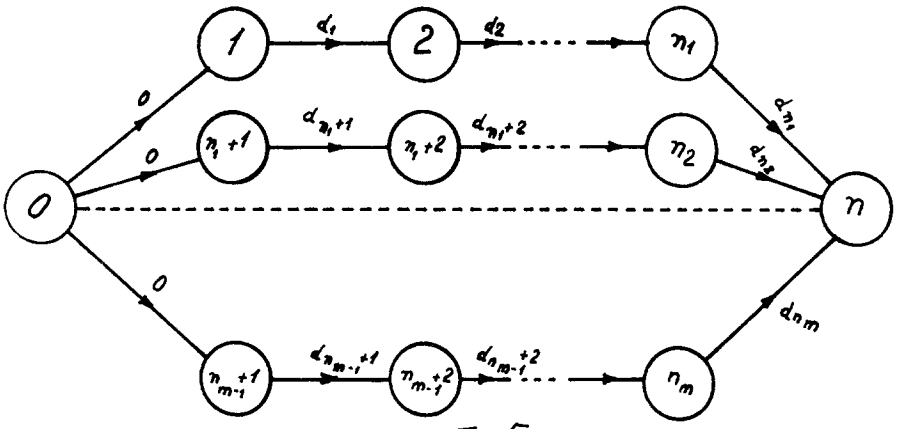

Fig. 5a

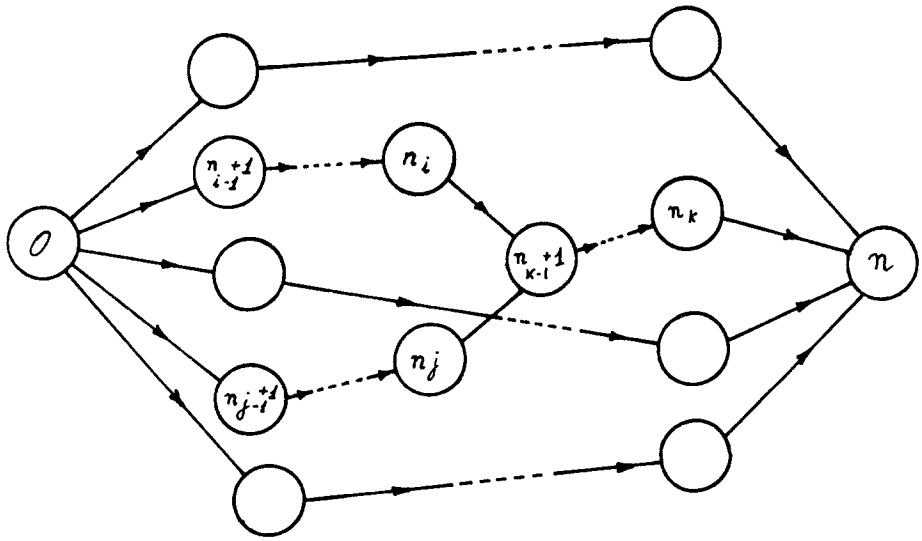

Fig. 5b

Figure 5

the sum $\sum d_j$ taken over all arcs of H will be called the length of the path. Any longest path in G will be called, as usually, a *critical path*. The problem (53), (54), and (55) is known to be equivalent to that of finding a critical path in G. When relations of the type (54′) are present, then G takes on the form shown in Fig. 5b.

On the other hand, the disjunctions (56) can be represented by disjunctive pairs of arcs, a concept introduced by B. Roy and B. Sussman.[35] With each disjunction $(t_j - t_h \geqq d_h) \vee (t_h - t_j \geqq d_j)$ we associate the disjunctive pair of arcs $[(h, j), (j, h)]$, defined as follows.

Two arcs of a graph form a disjunctive pair, if any path in the graph is permitted to meet at most one member of the pair. A graph containing disjunctive pairs of arcs will be termed a *disjunctive graph*.

Let X be the set of nodes and Q the set of arcs of G, i.e., let $G = (X,Q)$, and let $h \epsilon X, j \epsilon X, (h, j) \notin Q, (j, h) \notin Q$. Suppose we introduce into G the disjunctive pair of arcs $[(h, j),(j, h)]$, visualized as in Fig. 6. Note that the lengths of (h, j) and (j, h) need not be equal. Then the disjunctive graph obtained

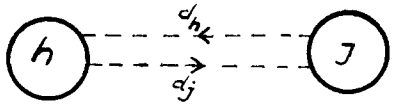

Figure 6

is equivalent to a disjunction between the two graphs $G' = [X, Q \cup (h, j)]$ and $G'' = [X, Q \cup (j, h)]$, i.e., a disjunction between the systems of relations (sets of constraints) expressed by G' and G''.

Accordingly, a disjunctive graph D obtained from the graph G by introducing p disjunctive pairs of arcs is equivalent to a disjunction between the 2^p members of the family $\mathcal{G}$ of graphs that can be obtained from G by adding to the set of its arcs exactly one arc of each of the p disjunctive pairs, in all possible combinations. We shall call G the *basic graph* for D.

Thus, we shall associate with the problem $\mathcal{P}$ a disjunctive graph D obtained from the PERT network G associated with (53), (54), and (55) by introducing disjunctive pairs of arcs between all pairs of nodes h, j such that $h \epsilon N^k$, $j \epsilon N^k$, for all $k \epsilon Q$, i.e., between all pairs of nodes representing the starting of operations to be carried out on the same machine.

Let W be the set of disjunctive pairs of arcs representing the constraints (56). Further, let $S_1, \cdots, S_u$ be the sets one can form by taking exactly one arc of each disjunctive pair $[(h, j), (j, h)] \epsilon W$, in all possible combinations, $(u = 2^p)$, and let $\{S_1, \cdots, S_u\} = S$. Then $\mathcal{G} = \{G_1, \cdots, G_u\}$, where $G_i = (X, Q \cup S_i)$, $S_i \epsilon S$.

A path, as well as a loop (defined as usually) in any one of the graphs $G_i \epsilon \mathcal{G}$, will be considered a path in D or a loop in D, accordingly.

Given a disjunctive pair of arcs $[(h,j), (j,h)]$, we shall call *normal* the arc representing the first term of the corresponding disjunction (56), and *inverse* the arc representing the second term. Let W^+ be the set of all normal arcs, and W^- the set of all inverse arcs of the pairs $[(h,j), (j,h)] \epsilon W$. Obviously, $W^+ \epsilon S$, $W^- \epsilon S$. Further, let $\mathcal{G}'$ be the set of those graphs $G_i \epsilon \mathcal{G}$ containing no loops. For each $G_i \epsilon \mathcal{G}'$ we shall denote by v_i the length of a critical path C_i in G_i. We shall say that C_k is a *minimaximal path in the disjunctive graph D*, if

$$v_k = \min_{G_i \epsilon \mathcal{G}'} v_i. \tag{57}$$

We have

THEOREM 4. *The problem $\mathcal{P}$ is equivalent to the problem of finding a set of arcs $S_k \epsilon S$ such that the graph $G_k = (X, Q \cup S_k)$ has no loops and that a critical path C_k in G_k is minimaximal in the disjunctive graph D. The length v_k of C_k yields the optimal value of t_n, while the optimal values of the variables t_j $(j = 1, \cdots, n_m)$ can be obtained by applying the critical path method to G_k.*

Proof. Each graph $G_i \epsilon \mathcal{G}$ corresponds to a problem—let us denote it $\mathcal{P}_i$—consisting of (53), (54), and (55) and exactly one constraint of each disjunctive pair in (56). Let $\mathcal{R} = \{\mathcal{P}_1, \cdots, \mathcal{P}_u\}$ be the set of all problems $\mathcal{P}_i$ that can be obtained from $\mathcal{P}$. A graph G_i having loops corresponds to a problem $\mathcal{P}_i \epsilon \mathcal{R}$ with no feasible solution. The conditions set by the theorem for the graph G_k translate therefore the requirement of finding, among all problems $\mathcal{P}_i \epsilon \mathcal{R}$, a problem $\mathcal{P}_k$ having a feasible solution and such that its optimal solution be minimal over the set of solutions to all problems $\mathcal{P}_i \epsilon \mathcal{R}$. But this is exactly what solving $\mathcal{P}$ amounts to.

Finding a Minimaximal Path in a Disjunctive Graph

The procedure we are going to describe is a specialization of Benders' partitioning method as described in the preceding section of this paper.

Let us replace the disjunctions (56) in $\mathcal{P}$ by pairs of constraints with 0-1 variables:

$$\begin{aligned} t_j - t_h + L y_{hj} &\geqq d_h, \\ -t_j + t_h - L y_{hj} &\geqq d_j - L, \qquad (h \epsilon N^k, j \epsilon N^k, k \epsilon Q) \quad (56') \\ y_{hj} &= 0 \text{ or } 1. \end{aligned}$$

Here L is a finite positive number sufficiently large for

$$d_h + t_h - t_j \leqq L \qquad (h \epsilon N^k, j \epsilon N^k, k \epsilon Q) \quad (58)$$

to hold for any values the variables t_h, t_j might take.

Then $\mathcal{P}$ becomes a mixed-variable zero-one problem that can be handled by the method described in the previous section. However, the two subproblems $\mathcal{P}^1$ and $\mathcal{P}^2$ derived from $\mathcal{P}$ display some specific features that offer a good ground for specializing that procedure.

TABLE II

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

Indeed, any assignment of a set of 0-1 values to the variables y_{hj} in $\mathcal{P}$ transforms (56′) into a set of constraints of the same type as (56) and defines a problem $\mathcal{P}_i \epsilon \mathfrak{R}$ of the type discussed in the proof to theorem 4. For, if $y_{hj}=0$ for a pair (h, j), then the first constraint of (56′) becomes $t_j - t_h \geqq d_h$, while the second one can be dropped because it simply restates (58). If $y_{hj}=1$, the second constraint becomes $t_h - t_j \geqq d_j$, while the first one can be dropped.

TABLE III

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

Therefore, our 'continuous' subproblem $\mathcal{P}^2$ will consist of (53), (54), and (55) and exactly one constraint of each disjunctive pair in (56). But this is a critical path problem, which is known to be infeasible whenever the associated graph contains a loop, and to have a finite optimal solution whenever the associated graph has no loops.

An optimal solution to the dual of such a problem is known to be a vector with 0-1 components, namely 1 for each arc in a critical path and 0 for each arc not in that path. In case there are loops, the dual has no finite optimum, but a vector with components 1 for the arcs in a loop, and 0 for the arcs not in that loop, is the direction-vector of an extreme ray of the cone associated with the solution-set to the dual.

Therefore, if $\mathcal{P}_r^1$ and $\mathcal{P}_r^2$ stand for the current subproblems $\mathcal{P}^1$ and $\mathcal{P}^2$, whenever the graph G_r associated with $\mathcal{P}_r^2$ has no loops, the solution of

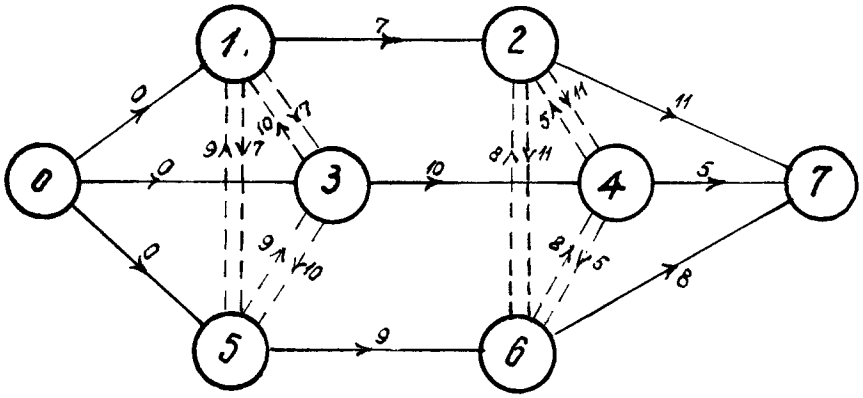

Figure 7

$\mathcal{P}_r^2$ generates for $\mathcal{P}_{r+1}^1$ (i.e., for the subproblem $\mathcal{P}^1$ of the next iteration) a constraint of the form

$$t_n + \sum_{(h,j)\in C_r \cap W^+} Ly_{hj} - \sum_{(h,j)\in C_r \cap W^-} Ly_{hj} \geqq v_r - L.\ |C_r \cap W^-|, \qquad (59)$$

where C_r is a critical path of length v_r in G_r, while $|C_r \cap W^-|$ is, obviously, the number of inverse arcs in C_r.

When G_r has loops, then we obtain for $\mathcal{P}_{r+1}^1$ a constraint of the form

$$\sum_{(h,j)\in C_r' \cap W^+} Ly_{hj} - \sum_{(h,j)\in C_r' \cap W^-} Ly_{hj} \geqq v_r' - L.\ |C_r' \cap W^-|, \qquad (60)$$

TABLE IV

r (No. of iteration)	t_7	y_{13}	y_{35}	y_{15}	y_{24}	y_{46}	y_{26}	$v_r - L\cdot\|C_r \cap W^-\|$
1	1	L	L					34
2		$-L$	$-L$	L				26-2L
3	1	$-L$	$-L$		L	L		50-2L
4					$-L$	$-L$	L	24-2L
5	1		$-L$		$-L$			35-2L
6	1		L		L		$-L$	43-L
7	1	$-L$					L	36-L
8	1		$-L$	L				31-L
8a	1					$-L$	L	31-L
9	1				L		$-L$	33-L
10	1		L				$-L$	38-L
11	1			$-L$	L	L		40-L

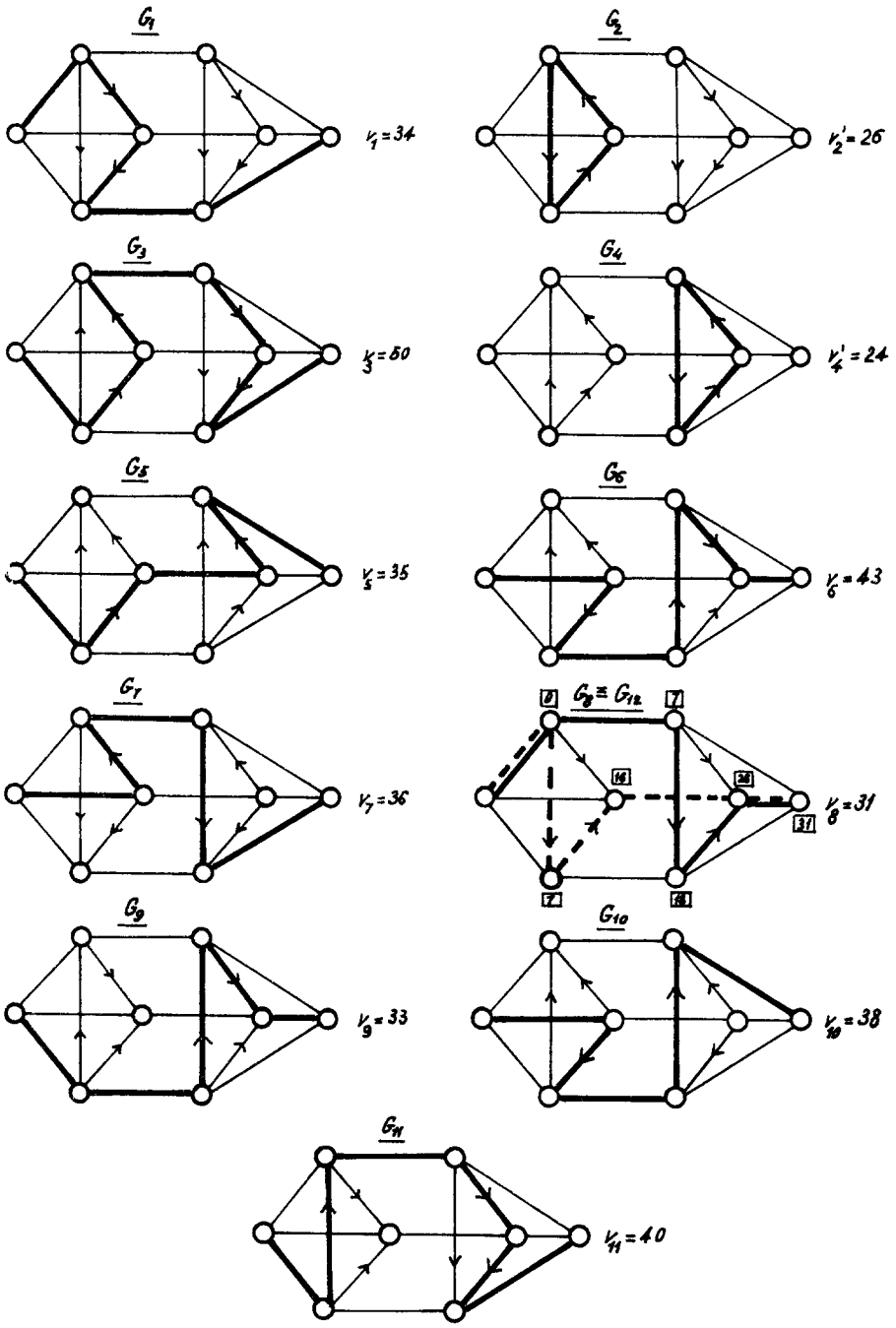

Figure 8

TABLE V

Machine	Item				
	1	2	3	4	5
1	2	3	1	4	0
2	3	2	3	0	0
3	0	3	0	3	4
4	0	0	2	1	4

where C_r' is a loop of length v_s' in G_r, and $|C_r' \cap W^-|$ is the number of inverse arcs in C_r'.

Thus, our procedure of solving $\mathcal{P}$ will run as follows. Two stages will be applied alternately. In stage A, we determine a set of arcs $S_r \epsilon S$ by solving the linear program $\mathcal{P}_r^1$ in the 0-1 variables y_{hj} and one continuous (nonnegative) variable t_n. In stage B we solve the linear program $\mathcal{P}_r^2$ in the continuous (nonnegative) variables t_j, t_n, which means that we find a loop (if there is one) or a critical path (if there is no loop) in the graph G_r associated with $\mathcal{P}_r^2$. If we find a critical path in G_r, a simple test will show whether it is or is not minimaximal in D. Each loop or critical path in G_r defines a new constraint for $\mathcal{P}^1$. Thus, if stage B does not find a minimaximal path in D, we go back to stage A and solve again $\mathcal{P}^1$, after having introduced the new constraint (or constraints) defined in stage B.

At the start (first iteration) we choose the set $S_0 = W^+$ and go to stage B.

Suppose we have executed r iterations and let us denote $\mathcal{P}_r^1$ and $\mathcal{P}_r^2$ the current problem $\mathcal{P}^1$ and $\mathcal{P}^2$ respectively. An iteration of the procedure can then be described as follows:

Stage A. Solve $\mathcal{P}_r^1$ with the method described in the preceding two sections. Let the optimal solution $(\bar{t}_n, \bar{y})$ to $\mathcal{P}_r^1$ define a set of arcs $S_r \epsilon S$ according to the rule

$$\begin{aligned} \bar{y}_{hj} = 0 &\Rightarrow (h, j) \epsilon S_r, \\ \bar{y}_{hj} = 1 &\Rightarrow (j, h) \epsilon S_r. \end{aligned} \tag{61}$$

REMARK. It often happens, especially at the first iterations, that some

TABLE VI

Operation	1	2	3	4	5	6	7	8	9	10	11	12	13
Item	1	1	2	2	2	3	3	3	4	4	4	5	5
Machine	1	2	1	3	2	1	2	4	1	4	3	4	3

of the variables y_{hj} can be assigned arbitrary values in the optimal solution. Therefore it is useful to amend (61) by the rule of assigning to each such arbitrary y_{hj} the value 0 (or the value 1).

Go to Stage B.

Stage B. Check whether $G_r = (X, Q \cup S_r)$ has any loops.

B.1. If G_r has no loops, find the critical paths of G_r and compute their length v_r.

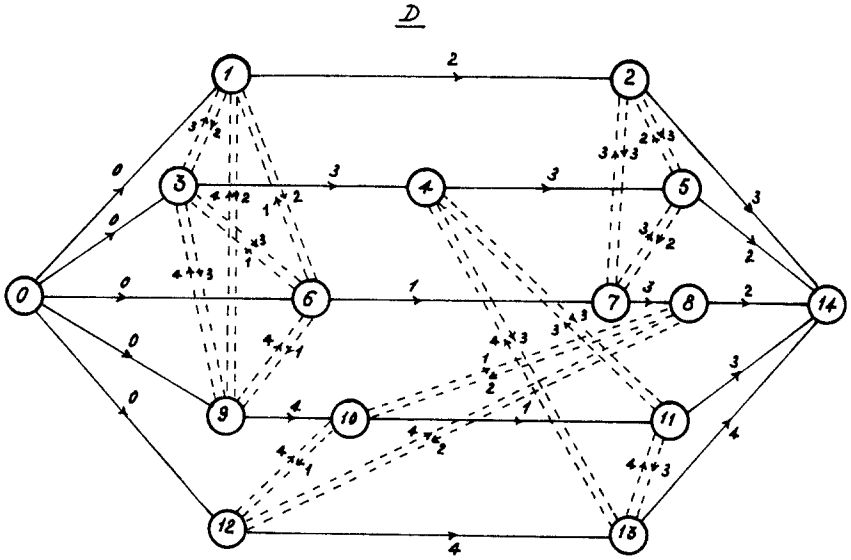

Figure 9

B.1.a. If $v_r = \bar{t}_n$, the critical path(s) we have found is (are) minimaximal in D, and together with the graph G_r it (they) yield(s) the optimal solution(s) to P. In this case the algorithm terminates.

B.1.b. If $v_r > \bar{t}_n$, then for each critical path C_r in G_r introduce into the problem $\mathcal{P}^1_{r+1}$ of the next iteration a constraint of the form (59) defined by C_r. Then go back to stage A.

B.2. If G_r has loops, then for each loop C_r' of length v_r' introduce into the problem $\mathcal{P}^1_{r+1}$ a constraint of the form (60) defined by C_r'. Then go back to stage A.

We have

THEOREM 5. *In a finite number of iterations, the above described method yields an optimal solution to* $\mathcal{P}$.

Proof. $\mathcal{P}$ always has a finite optimal solution, and in the above discussion it has been shown that our procedure is a specialization of Benders' partitioning method that is known to end in a finite number of iterations.

Numerical Examples

Example 1. Three items (lots of items) are to be processed on two machines. The time required for the processing of each item on each machine is shown in Table II, while Table III shows the indexing of the operations.

The associated disjunctive graph D is shown in Fig. 7.

The problem is solved in 12 iterations Table IV contains the coefficients of the constraints of $\mathcal{P}_r^1$ generated at each iteration except the last

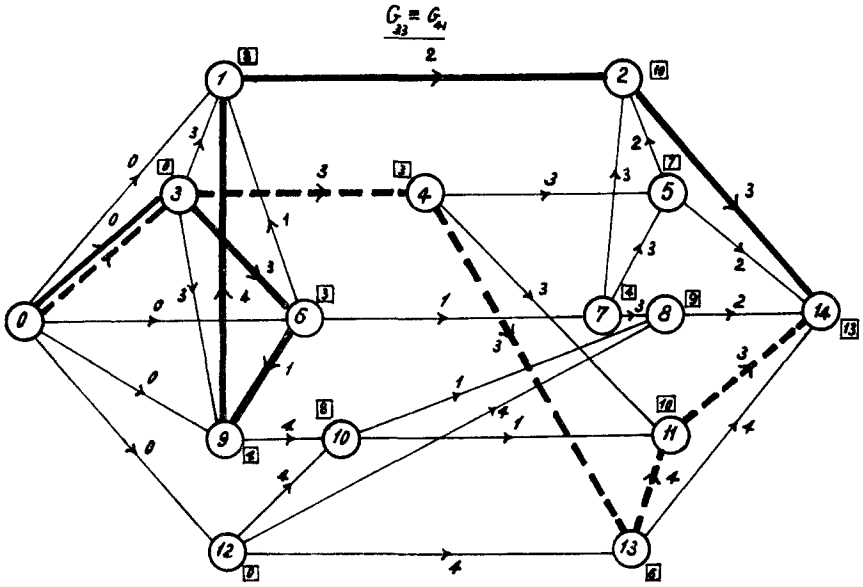

Figure 10

one. (At each iteration one constraint is generated, except for iteration 8, when there are two.) We have taken $L=50$.

The sequence of problems $\mathcal{P}_r^2$ $(r=1, \cdots, 11)$ is shown in Fig. 8, which contains the graphs G_r $(r=1, \cdots, 11)$ with their critical paths (or loops). The graph G_{12} generated at the last iteration is the same as G_8, the only one containing two critical paths. The optimal solution is

$$\bar{t}_1=0, \quad \bar{t}_2=7, \quad \bar{t}_3=16, \quad \bar{t}_4=26, \quad \bar{t}_5=7, \quad \bar{t}_6=18, \quad \bar{t}_7=31.$$

It has been found after constructing 11 of the $2^6=64$ graphs $G_i \in \mathcal{G}$.

Example 2. This example is taken from reference 34. Five items (lots of items) are to be processed on four machines. The time intervals needed for processing the various items on various machines are given in Table V, while Table VI shows the indexing of operations.

The disjunctive graph D, associated with the problem is shown in Fig. 9. It contains 15 disjunctive pairs of arcs, i.e., there are $2^{15}=32.768$ graphs $G_i \epsilon \mathcal{G}$.

The problem is solved in 41 iterations, i.e., the graph G_r yielding the optimal solution (shown in Fig. 10) is found after constructing 40 of the 32.768 graphs G_i. The optimal solution is

$$\bar{t}_1=8,\ \bar{t}_2=10,\ \bar{t}_3=0,\ \bar{t}_4=3,\ \bar{t}_5=7,\ \bar{t}_6=3,\ \bar{t}_7=4,\ \bar{t}_8=9,\ \bar{t}_9=4,\ \bar{t}_{10}=8.$$

$$\bar{t}_{11}=10,\quad \bar{t}_{12}=0,\quad \bar{t}_{13}=6,\quad \bar{t}_{14}=13.$$

REFERENCES

1. E. Balas, "Programare liniară cu variabile bivalente" ("Linear Programming with Zero-One Variables"), *Proc. Third Scientific Session on Statistics, Bucharest*, December 5–7, 1963.
2. ———, "Un algorithme additif pour la résolution des programmes linéaires en variables bivalentes," *Comptes Rendus de l'Académie des Sciences* (Paris) **258,** 3817–3820 (April 1964).
3. ———, "An Additive Algorithm for Solving Linear Programs with Zero-One Variables," *Opns. Res.* **13,** 517–546 (1965).
4. ———, "Extension de l'algorithme additif à la programmation en nombres entiers et à la programmation non linéaire," *Comptes Rendus de l'Académie des Sciences* (Paris) **258,** 5136–5139 (May 1964).
5. ———, "La Méthode du filtre (l'algorithme additif accéléré) pour la résolution des programmes linéaires en variables bivalentes," *Comptes Rendus de l'Académie des Sciences* (Paris) **262,** 766–769 (March 1966).
6. ———, "Adaptation de l'algorithme additif accéléré à la programmation mixte," *Comptes Rendus de l'Académie des Sciences* (Paris) **262,** 831–834 (April 1966).
7. G. B. Dantzig, "On the Significance of Solving Linear Programming Problems with Some Integer Variables," *Econometrica* **28,** 30–44 (1960).
8. F. Glover, "A Bound Escalation Method for the Solution of Integer Linear Programs," *Cahiers du Centre d'Etudes de Recherche Opérationnelle* (Brussels) **6** (1964).
9. ———, "A Multiphase-Dual Algorithm for the Zero-One Integer Programming Problem," *Opns. Res.* **13,** 879–919 (1965).
10. B. B. Gordon, R. W. Hose, A. P. Lechler, L. D. Nelson, and T. Radó, "Computer Studies of a Certain Class of Integer Linear Programs," Battelle Memorial Research Institute, Columbus, Ohio, 1964.
11. P. L. Ivanescu, "Pseudo-Boolean Programming and Applications" (Abstract of Doctor's Thesis), *Lecture Notes in Mathematics*, 1965, No. 9, Springer Verlag, Berlin-Heidelberg, New York.
12. E. M. L. Beale and R. E. Small, "Mixed Integer Programming by a Branch and Bound Technique," IFIP Congress 1965, New York.
13. M. L. Balinski, "Integer Programming: Methods, Uses, Computation," *Management Sci.* **12,** 253–313 (1965).

14. B. Bouvier and G. Messoumian, *Programmes linéaires en variables bivalentes (Algorithme de Balas)*, Faculté des Sciences, Université de Grenoble, 1965.
15. R. J. Freeman, "Computational Experience with a 'Balasian' Integer Programming Algorithm," *Opns. Res.* **14**, 935–940 (1966).
16. C. C. Petersen, *Selection of New Research and Development Opportunities in Light of Budget Constraints*, The Arizona State University, 1965.
17. P. Bertier and P. T. Nghiem, *Résolution de problèmes en variables bivalentes (Algorithme de Balas et procédure SEP)*, SEMA, Paris, Note D.S. No. 33, 1965.
18. F. Glover and S. Zionts, "A Note on the Additive Algorithm of Balas," *Opns. Res.* **13**, 546–549 (1965).
19. A. Geoffrion, "Integer Programming by Implicit Enumeration and Balas' Method," *SIAM Rev.* **9**, 178–190 (1967).
20. A. J. Goldman and A. W. Tucker, "Theory of Linear Programming," *Linear Inequalities and Related System*, H. W. Kuhn and A. W. Tucker (eds.), Princeton University Press, Princeton, 1956.
21. J. D. C. Little, K. G. Murty, D. W. Sweeney, and C. Karel, "An Algorithm for the Traveling Salesman Problem," *Opns. Res.* **11**, 972–989 (1963).
22. F. Radó, "Programare liniară cu conditii logice" ("Linear Programming with Logical Conditions"), Comunicările Academiei RPR **13**, 1039–1041 (1963).
23. P. Bertier and B. Roy, "Procédure de résolution pour une classe de problèmes pouvant avoir un caractere combinatoire," *Cahiers du Centre d'Etudes de Recherche Opérationnelle* (Bruxelles) **6**, 202–208 (1964).
24. G. B. Dantzig, "Upper Bounds, Secondary Constraints and Block Triangularity in Linear Programming," *Econometrica* **23**, 174–183 (1955).
25. J. F. Benders, "Partitioning Procedures for Solving Mixed-Variables Programming Problems," *Numerische Mathematik* **4**, 238–252 (1962).
26. C. Berge, *The Theory of Graphs and its Applications*, Methuen-Wiley, 1962 (English Translation).
27. B. Fleischmann, "Computational Experience with the Additive Algorithm of Balas," *Opns. Res.* **15**, 153–154 (1967).
28. S. M. Johnson, "Optimal Two- and Three-Stage Production Schedules with Setup Times Included," *Naval Res. Log. Quart.* **1**, 61–68 (1954).
29. F. H. Bowman, "The Schedule-Sequencing Problem," *Opns. Res.* **7**, 621–624 (1959).
30. G. B. Dantzig, "A Machine-Job Scheduling Model," *Management Sci.* **6**, 191–196 (1960).
31. A. S. Manne, "On the Job-Shop Scheduling Problem," *Opns. Res.* **8**, 219–223 (1960).
32. G. Giffler and G. L. Thompson, "Algorithms for Solving Production Scheduling Problems," *Opns. Res.* **8**, 487–503 (1960).
33. A. L. Lurye, "O nekotorîh zadaciah kalendarnovo planirovania," *Problemî Kibernetiki* **7**, 201–208 (1962).
34. L. Németi, "Das Reihenfolgeproblem in der Fertigungsprogrammierung und Linearplanung mit logischen Bedingungen," *Mathematika* **6(29)**, 87–92 (1964).

35. B. Roy and B. Sussman, "Les problèmes d'ordonnancement avec contraintes disjonctives," SEMA, Note D.S. No. 9 bis (déc. 1964).
36. E. Balas, "Teoria grafurilor şi încărcarea optima a utilajelor la fabricile de mobilă" ("Graph Theory and Optimal Machine Loading in the Manufacturing of Furniture"), *Proc., Scientific Session of the Forestry Research Institute, Bucharest, June 1966.*
37. ———, "Finding a Minimaximal Path in a Disjunctive PERT Network," *Théorie des Graphes. Journées Internationales d'Études, Rome 1966*, Dunod, Paris, 1967.
38. E. L. Lawler and D. E. Wood, "Branch-and-Bound Methods: A Survey," *Opns. Res.* **14,** 699–719 (1966).
39. ——— and M. D. Bell, "A Method for Solving Discrete Optimization Problems," *Opns. Res.* **14,** 1098–1112 (1966).
40. W. D. James, "Some Remarks on a Numerical Example of Balas and on the Note by Glover and Zionts," *Opns. Res.* **14,** 941–942 (1966).
41. K. M. Brauer, "A Note on the Efficiency of Balas' Algorithm," forthcoming in *Opns. Res.*
42. P. C. Gilmore and R. E. Gomory, "The Theory and Computation of Knapsack Functions," *Opns. Res.* **14,** 1045–1074 (1966).
43. C. C. Petersen, "Computational Experience with Variants of the Balas Algorithm Applied to the Selection of R & D Projects," forthcoming in *Management Sci.* **13,** 736–750 (1967).
44. E. Balas, "A Note on the Branch-and-Bound Principle," University of Toronto, February 1967.
45. ———, *Discrete Programming by Implicit Enumeration*, forthcoming in the series "Lecture Notes in Mathematics," Springer, 1967.
46. E. Balas, *Duality and Pricing in Integer Programming*, Operations Research House, Stanford University, July 1967.
47. B. Fleischmann, "Lösungsverfahren und Anwendungen der ganzzahligen linearen Optimierung," Thesis, Hamburg, 1967.
48. C. E. Lemke and K. Spielberg, *Direct Search Zero-One and Mixed-Integer Programming*, IBM, New York Scientific Center, June, 1966.
49. S. Woiler, *Implicit Enumeration Algorithms for Discrete Optimization Problems*, Department of Industrial Engineering, Stanford University, May 1967.