Operations Research

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

An Additive Algorithm for Solving Linear Programs with Zero-One Variables

Egon Balas,

To cite this article:
Egon Balas, (1965) An Additive Algorithm for Solving Linear Programs with Zero-One Variables. Operations Research 13(4):517-546. https://doi.org/10.1287/opre.13.4.517

Operations Research

July-August 1965

AN ADDITIVE ALGORITHM FOR SOLVING LINEAR PROGRAMS WITH ZERO-ONE VARIABLES†

Egon Balas

Centre of Mathematical Statistics, Rumanian Academy, Bucharest

(Received March 2, 1964)

An algorithm is proposed for solving linear programs with variables constrained to take only one of the values 0 or 1. It starts by setting all the n variables equal to 0, and consists of a systematic procedure of successively assigning to certain variables the value 1, in such a way that after trying a (small) part of all the 2^n possible combinations, one obtains either an optimal solution, or evidence of the fact that no feasible solution exists. The only operations required under the algorithm are additions and subtractions; thus round-off errors are excluded. Problems involving up to 15 variables can be solved with this algorithm by hand in not more than 3–4 hours. An extension of the algorithm to integer linear programming and to nonlinear programming is available, but not dealt with in this article.

IT IS well known that important classes of economic (and not only economic) problems find their mathematical models in linear programs with integer variables. Prominent among these problems are those that correspond to linear programs with variables taking only one of the values 0 or 1. A rapidly growing literature on the subject describes many practical instances of such problems in a large variety of fields.‡

At present, several methods are available for solving linear programs of the type discussed here.§ Best known among them are R. E. GOMORY's algorithms,[13,14] for solving linear programs with integer variables. They use the dual simplex method and impose the integer conditions by adding

† Paper presented at the International Symposium on Mathematical Programming, July 1964, London.

‡ *See* references 1–4; reference 5 (pp. 650–656, 695–700); reference 6; reference 7 (pp. 194–202); reference 8 (pp. 535–550); references 9–11.

§ *See* references 12–27; reference 5 (pp. 700–712); reference 7 (pp. 160–194); reference 8 (pp. 514–535); reference 28 (pp. 190–205).

automatically generated new constraints to the original constraint set. These are satisfied by any integer solution to the latter, but not by the solution reached at the stage of their introduction. The cutting-plane approach has also been used by E. M. L. BEALE[17] and R. E. Gomory[14] to develop algorithms for solving the mixed case, when some but not all of the variables are bound to be integers. The procedures of references 6 and 24 also belong to this family.

Another type of algorithm for integer (and mixed integer) linear programs, developed by A. H. LAND AND A. G. DOIG,[18] also starts with a noninteger optimal solution and then finds the optimal integer (or mixed-integer) solution through systematic parallel shifts of the objective function-hyperplane. The methods of references 21, 22, 25, and 26 also come under this heading.

A different approach to the problem was initiated by R. FORTET[29] on the lines of Boolean algebra, and continued by R. CAMION[30] with the introduction of Galois fields. On these lines, P. L. IVANESCU[23] developed an algorithm for solving discrete polynomial programs.

The algorithm proposed in this paper† represents a *combinatorial* approach to the problem of solving discrete-variable linear programs in general, and linear programs with zero-one variables in particular. As an abbreviated enumeration procedure, it is kindred in conception to combinatorial methods developed in related areas (*see*, for instance, references 33 and 34). This algorithm is first of all a direct method for solving linear programs with zero-one variables, and for this particular type of problem it seems to work very efficiently. It has also been extended[32] to linear programs with integer variables, and, as a method of approximation, to nonlinear programs of a more general type than those usually dealt with.

BASIC IDEAS AND OUTLINE OF THE ADDITIVE ALGORITHM

THE GENERAL form of a linear program with zero-one variables may be stated as follows:

Find x' minimizing (maximizing)

$$c'x', \qquad (1')$$

subject to

$$A'x' \gtreqless b^i, \qquad (2')$$

$$x_j' = 0 \text{ or } 1, \qquad (j \in N) \quad (3')$$

† An initial version of the additive algorithm was presented at the Third Scientific Session on Statistics, Bucharest, December 5–7, 1963 (*see* reference 9). A brief note on this algorithm was also published in reference 31, and another one on its extensions in reference 32. For an interesting graph-theoretical interpretation of the additive algorithm and a comparison with some later developments *see* reference 35.

where $x' = (x_j')$ is an n-component column-vector, $c' = (c_j')$ is a given n-component row-vector, $A' = (a_{ij}')$ is a given $q \times n$ matrix and $b' = (b_j')$ is a given q-component column-vector, while $\{1, \cdots, q\} = Q$, $\{1, \cdots, n\} = N$.

However, we wish to consider the problem in a slightly different form, namely with all the constraints being inequalities of the same form $\leqq$, and all the coefficients of the objective function (to be minimized) being nonnegative. Any problem of the type (1′), (2′), (3′) can be brought to this form by the following operations:

(a). Replacing all equations by two inequalities.

(b). Multiplying by -1 all inequalities of the form $\geqq$.

(c). Setting

$$x_j = \begin{cases} x_j' & \text{for } c_j' \geqq 0 \text{ when minimizing; for } c_j' \leqq 0 \text{ when maximizing,} \\ 1 - x_j' & \text{for } c_j' < 0 \text{ when minimizing; for } c_j' > 0 \text{ when maximizing.} \end{cases}$$

Following this and the introduction of an m-component nonnegative slack vector y, the problem may be restated thus:

Find x such that

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

subject to

$$Ax + y = b, \tag{2}$$

$$x_j = 0 \text{ or } 1, \qquad (j \epsilon N) \tag{3}$$

$$y \geqq 0, \tag{4}$$

where $c \geqq 0$, and where x, c, A, and b are to be obtained from x', c', A' and b' through the above described transformation. The dimension of x and c remains n. Let b be m-dimensional, with $\{1, \cdots, m\} = M, m \geqq Q$.

The problem (1), (2), (3), and (4) will be labeled P.

Let a_j stand for the jth column of A.

An $(n+m)$-dimensional vector $u = (x,y)$ will be called a *solution*, if it satisfies (2) and (3); a *feasible* solution, if it satisfies (2), (3), and (4); and an *optimal* (feasible) solution, if it satisfies (1), (2), (3), and (4).

Let us denote P^s the linear program defined by (1), (2), and (4), and the constraints

$$x_j \geqq 0, \qquad (j \epsilon N) \tag{3a}$$

$$x_j = 1, \qquad (j \epsilon J_s) \tag{3b_s}$$

where J_s is a subset of N. Let $J_0 = \phi$, and thus P^0 be the problem defined by (1), (2), (3a), and (4).

The fundamental idea underlying our algorithm runs on the following lines. We start with the ordinary linear program P^0 with $u^0 = (x^0, y^0) = (0, b)$, which is obviously a dual-feasible solution to P^0 (because $c \geqq 0$). The corresponding basis consists of the unit-matrix $I_{(m)} = (e_i)$ $(i = 1, \cdots, m)$, e_i being the ith unit vector. For some i such that $y_i^0 < 0$ we choose, accord-

ing to a certain rule, a vector a_{j_1} such that $a_{ij_1}<0$, to be introduced into the basis. But instead of introducing a_{j_1} in place of a vector e_i in the basis, as would be the case in the dual simplex method, we add to P^0 the constraint $x_{j_1}=1$, in the slightly modified form $-x_{j_1}+y_{m+1}=-1$, where y_{m+1} is an artificial variable. Thus we obtain the problem P^1 as defined above, with $J_1=\{j_1\}$, i.e., the problem consisting of (1), (2), (3a), (4), and the additional constraint

$$x_{j_1}=1. \tag{3b$_1$}$$

It is easy to see that the set $x_j=0(j\epsilon N)$, $y_i=b_i(i\epsilon M)$, $y_{m+1}=-1$, is a dual-feasible solution to P^1. In the extended basis $I_{(m+1)}=(e_i)(i=1,\cdots,m+1)$, the $(m+1)$st unit vector e_{m+1} corresponds to y_{m+1}. It is in the place of this unit vector e_{m+1} that we introduce a_{j_1}, and thus x_{j_1} takes the value 1 in the new solution to P^1 that obviously remains dual-feasible. As the artificial variable y_{m+1}, which becomes 0, does not play any role henceforth, it may be abandoned and the new solution may be written $u^1=(x^1,y^1)$ $=(x_1^1,\cdots,x_n^1,y_1^1,\cdots,y_m^1)$.

Given the particularly simple form of the additional constraint, the pivot around the element -1 consists in fact of the algebraic addition $b-a_{j_1}$. Thus, the new dual-feasible solution $u^1=(x^1,y^1)$ to P^1 is

$$x_j^1=\begin{cases}1(j=j_1),\\ 0(j=N-\{j_1\}),\end{cases}\qquad y_i^1=b_i-a_{ij_1}. \qquad (i\epsilon M)$$

As the operations to be carried out at each iteration consist solely of such additions and subtractions, we call the algorithm *additive*.

If the solution-vector u^1 still has negative components, then according to the above mentioned rules we choose another vector a_{j_2} to be introduced into the basis, and we add to P^1 the new constraint $x_{j_2}=1$, in the form $-x_{j_2}+y_{m+2}=-1$, y_{m+2} being another artificial variable. This yields the problem P^2, consisting of (1), (2), (3a), and (4) and the additional constraint set $(3b_2)$, made up of $x_{j_1}=1$, $x_{j_2}=1$. The set $x_{j_1}=1$, $x_j=0[j\epsilon(N-\{j_1\})]$, $y_i=b_i-a_{ij_1}(i\epsilon M)$, $y_{m+2}=-1$, is a dual-feasible solution to P^2. The vector a_{j_2} is now introduced in place of e_{m+2}, and x_{j_2} takes the value 1 in the new solution to P^2, which obviously remains dual-feasible.

As the artificial variable y_{m+2} does not play any role henceforth (remaining all the time equal to 0), it may be dropped (as was the case for y_{m+1}), and the new dual-feasible solution to P^2 is $u^2=(x^2,y^2)$, where

$$x_j^2=\begin{cases}1(j=j_1,j_2),\\ 0[j\epsilon(N-\{j_1,j_2\})],\end{cases}\qquad y_i^2=y_i^1-a_{ij_2}. \qquad (i\epsilon M)$$

This procedure is repeated until either a solution u^s is reached with all components nonnegative, or evidence is obtained that such a solution to P^s does not exist.

If a nonnegative vector $u^s = (x^s, y^s)$ is obtained, it is an optimal (feasible) solution to P^s. Such a solution may or may not be optimal for P, but it is always a feasible solution to P.

The procedure is then started again from a solution $u^p (p < s)$ chosen according to certain rules, with other rules governing the choice of vectors to be introduced into the basis, until either another feasible solution u^t such that $z_t < z_s$ is obtained (z_p being the value of z for u^p) $(p = 0, 1, \cdots)$, or evidence is obtained of the absence of such solutions.

The sequence $u^q (q = 0, 1, \cdots)$ converges towards an optimal solution.

This procedure might also be called a pseudo-dual algorithm, because, as in the dual simplex method, it starts with a dual-feasible solution and then successively approaches the primal-feasible 'region,' safeguarding at all times the property of dual-feasibility. However, a real dual simplex iteration never takes place; the dual simplex criterion for choosing the vector to enter the basis is not used, nor are any of the vectors $e_i (i = 1, \cdots, m)$ ever 'eliminated' from the basis in the sense of being replaced by another vector. All changes in the basis occur through adding new unit-rows and unit-columns or dropping them and, what is most important, the coefficient-matrix A remains unchanged. As the new unit-rows and unit-columns introduced at each iteration never play any role in a further iteration, they need neither be retained nor explicitly written out.

The nature of the additive algorithm will best be pointed out by a comparison with what would have to be done if we were to try all the existing solutions. As the variables of P can only take the values 0 or 1, the set $U = \{u\}$ of all solutions to (2) and (3) is of course finite, and the number of elements in U is 2^n, where n is the number of variables x_j. The process of trying all the 2^n possible combinations (i.e., solutions) is illustrated for a problem with 5 variables in Fig. 1. The figure has 5 'levels.' Starting from the top, at each 'level' we assign in turn the values 0 and 1 to the variable bearing the number of the level. All points along a line running down to the left represent one and the same solution, while all points along a line running down to the right represent different solutions. By continuing this procedure until all the variables have been assigned the values 0 and 1, we obtain the whole set of $2^5 = 32$ solutions.

What our additive algorithm amounts to is, in fact, a set of rules according to which one may obtain an optimal solution (if such a solution exists) by following *some* branches of the tree in Fig. 1, while neglecting most of them. Starting from a situation in which all n variables are equal to 0, our algorithm consists of a systematic procedure of assigning the value 1 to some of the variables in such a way that after trying a small part of all the 2^n possible combinations, one obtains either an optimal solution, or evidence of the fact that no feasible solution exists. This is achieved

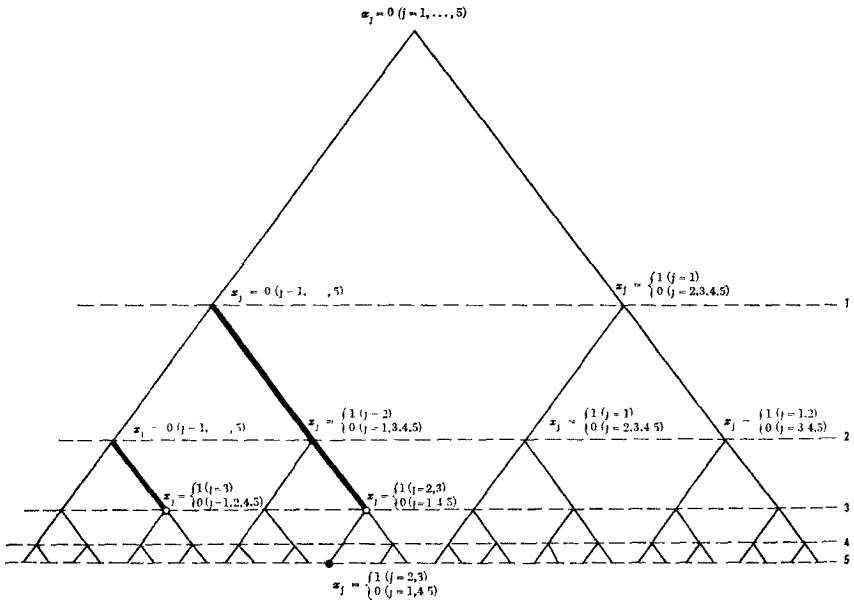

Figure 1

through a set of rules determining at each interation (a) a subset of variables that are candidates for being assigned the value 1; (b) the variable to be chosen among the candidates. At certain stages of the procedure it becomes clear that either an optimal solution has been obtained, or there is no optimal solution with value 1 for all the variables that had been assigned this value. The procedure is then stopped and started again from a previous stage. In other words, the rules of the algorithm identify such branches of the solution-tree, which may be abandoned because they cannot lead to a feasible 'solution' better than the one already obtained. Thus, in Fig. 1, which illustrates our first numerical example presented in the final section, only the thick lines are to be followed and the corresponding solutions to be tested. This means that, instead of all $2^5=32$ solutions, only the following 3 had to be tried:

$$u^0 \quad \text{with} \quad x_j^0=0, \qquad (j=1, \cdots, 5)$$

$$u^1 \quad \text{with} \quad x_j^1=\begin{cases}1, & (j=3)\\ 0, & (j=1,2,4,5)\end{cases}$$

$$u^2 \quad \text{with} \quad x_j^2=\begin{cases}1, & (j=2,3)\\ 0, & (j=1,4,5)\end{cases}$$

and this latter solution has been found to be optimal. The 'stop signals'

of the algorithm (represented by the circles at the end of the thick lines) make sure that no feasible solution 'better' than the one obtained exists beyond them.

Of course, under such circumstances, the efficiency of the algorithm depends largely on the efficiency of these 'stop signals' i.e., on the number of branches that need not be followed. As it will be shown later in greater detail, in most cases the algorithm succeeds in reducing the subset of solutions to be tested to a relatively small fraction of the complete set.

SOME DEFINITIONS AND NOTATIONS

LET US consider problem P.

As each constraint of the set (2) contains exactly one component of $\boldsymbol{y}$, a solution $\boldsymbol{u}^p=(\boldsymbol{x}^p,\boldsymbol{y}^p)$ is uniquely determined by the set $J_p=\{j|j\epsilon N, x_j^p=1\}$. For, if

$$x_j^p=\begin{cases}1(j\epsilon J_p),\\ 0[j\epsilon(N-J_p)],\end{cases} \tag{5}$$

then

$$y_i^p=b_i-\sum_{j\epsilon J_p} a_{ij}. \qquad (i\epsilon M) \tag{6}$$

As already shown, the additive algorithm generates a sequence of solutions. We shall denote the sth term of this sequence by

$$\boldsymbol{u}^s=\boldsymbol{u}(j_1,\cdots,j_r)=(\boldsymbol{x}^s,\boldsymbol{y}^s), \tag{7}$$

where

$$\{j_1,\cdots,j_r\}=J_s=\{j|j\epsilon N, x_j^s=1\}, \tag{8}$$

while z_s will represent the value of the form (1) for $\boldsymbol{u}^s$.

The sequence starts with $\boldsymbol{u}^0$, for which $J_0=\phi$, i.e., $\boldsymbol{x}^0=\boldsymbol{0}$, $\boldsymbol{y}^s=\boldsymbol{b}$, and $z_0=0$. Of course, $J_0\subset J_p$ for any $p\neq 0$.

The set of values taken by the objective function for the feasible solutions obtained until iteration $\boldsymbol{s}$ will be denoted

$$Z_s=\{z_p|p\leqq s, u^p\geqq 0\}. \tag{9}$$

If this set is not void, its smallest element will be called the *ceiling* for $\boldsymbol{u}^s$. If it is void, the rôle of the ceiling will be performed by ∞. Thus, we shall denote the ceiling for $\boldsymbol{u}^s$

$$z^{*(s)}=\begin{cases}\infty & \text{if } Z_s=\phi,\\ \min_{Z_s} z_p & \text{if } Z_s\neq\phi.\end{cases} \tag{10}$$

At each iteration $s+1$, the new vector to be introduced into the basis will be chosen from a subset of $\{\boldsymbol{a}_j|j\epsilon N\}$, called *the set improving vectors for the solution* $\boldsymbol{u}^s$. We shall denote by N_s the corresponding set of indices j (of course, $N_s\subseteq N$), and we shall define it more precisely below.†

† Throughout this paper the symbol $\subseteq$ will be used for inclusion, while $\subset$ will stand for strict inclusion.

We now define certain values which will serve as a criterion for the choice of the vector to be introduced into the basis. Thus, for each solution u^s and for each $j \epsilon N_s$, we define the values

$$v_j^s = \begin{cases} \sum_{i \epsilon M_j^{s-}} (y_i^s - a_{ij}), & (j \epsilon N_s ; M_j^{s-} \neq \phi) \\ 0, & (j \epsilon N_s ; M_j^{s-} = \phi) \end{cases} \quad (11)$$

where

$$M_j^{s-} = \{i | y_i^s - a_{ij} < 0\}. \quad (12)$$

The meaning of these values is obvious: $v_{j_s}^s$ is the sum of negative components of the solution-vector u^{s+1}, which can be obtained from the solution-vector u^s by setting $J_{s+1} = J_s \cup \{j_s\}$.

As already said, the values v_j^s are to serve as a criterion for the choice of the new vector to be introduced into the basis. This criterion has been found efficient; however, it must be emphasized that it is empirically chosen, being neither compelling, nor essential for the additive algorithm, which may do as well with some other criterion. The choice of the criterion for introducing a new vector into the basis may of course heavily affect the efficiency of the algorithm, but has no influence on its finiteness.

Under the additive algorithm, the values v_j^k assigned to a certain solution u^k are successively cancelled in the subsequent iterations according to certain rules. Let $C_k^s (k \leqq s)$ stand for the set of those j for which the values v_j^k assigned to the solution u^k have been cancelled before the solution u^s has been obtained. ($C_k^k = \phi$ by definition.)

The set of those j for which the values v_j^p assigned to any one of the solutions u^p such that $p < s$ and $J_p \subset J_s$ have been cancelled before obtaining the solution u^s, will be denoted

$$C^s = \bigcup_{p | J_p \subset J_s} C_p^s. \quad (13)$$

We shall now define for the solution u^s the set of those $j \epsilon (N - C^s)$ such that, if a_j were introduced into the basis so as to yield $J_{s+1} = J_s \cup \{j\}$, the value of the objective function would hit the ceiling for u^s:

$$D_s = \{j | j \epsilon (N - C^s), \quad c_j \geqq z^{*(s)} - z_s\}. \quad (14)$$

Further, the set of those $j \epsilon [N - (C^s \cup D_s)]$ will be defined such that, if a_j were introduced into the basis so as to yield $J_{s+1} = J_s \cup \{j\}$, no negative y_i^s would be increased in value:

$$E_s = \{j | j \epsilon [N - (C^s \cup D_s)], \quad y_i^s < 0 \Rightarrow a_{ij} \geqq 0\}. \quad (15)$$

We are now in a position to give a proper definition to the set of improving vectors for a solution u^s. It is the set of those a_j, for which j belongs to

$$N_s = N - (C^s \cup D_s \cup E_s). \quad (16)$$

Obviously, $N_s = \phi$ for any feasible solution u^s.

Similarly to D_s, we define for the pair of solutions u^k and $u^s(k<s)$ the set of those $j\epsilon(N_k-C_k^s)$ such that, if a_j were introduced into the basis so as to yield $J_{s+1}=J_k\cup\{j\}$, then z_{s+1} would hit the ceiling for u^s:

$$D_k^s=\{j|j\epsilon(N_k-C_k^s),\qquad c_j\geqq z^{*(s)}-z_k\}. \tag{17}$$

Finally, given a pair of solutions u^s and u^k, such that $u^s=u(j_1,\cdots,j_r)$, $u^k=u(j_1,\cdots,j_{r-h})$, $(1\leqq h\leqq r; J_k\subset J_s)$, we define *the set of improving vectors for the solution u^k, left after iteration s.* It is the set of those a_j, for which j belongs to

$$N_k^s=N_k-(C_k^s\cup D_k^s). \tag{18}$$

The sets defined above by (16) and (18) will play a central role in our algorithm. Whenever a solution u^s is reached, only the improving vectors for that solution are considered for introduction into the basis. Whenever the set of improving vectors for a solution u^s is found to be void, this is to be interpreted as a 'stop signal,' which means that there is no feasible solution u^t such that $J_s\subset J_t$ and $z_t<z^{*(s)}$. In such cases we have to take up our procedure from a previous solution u^k, to be identified according to certain rules, and for any such solution only the set of improving vectors for that solution u^k left after iteration s is to be considered for introduction into the basis.

STATEMENT OF THE ADDITIVE ALGORITHM

We start with the dual-feasible solution u^0, for which

$$x^0=0,\qquad y^0=b\quad\text{and}\quad z_0=0. \tag{19}$$

Let us suppose that after s iterations we have obtained the solution $u^s=u(j_1,\cdots,j_r)$, for which

$$x_j^s=\begin{cases}1, & (j\epsilon J_s)\\ 0, & [j\epsilon(N-J_s)]\end{cases} \tag{20}$$

$$y_i^s=b_i-\sum_{j\epsilon J_s}a_{ij},\qquad (i\epsilon M)$$

and

$$z_s=\sum_{j\epsilon J_s}c_j. \tag{21}$$

The following procedure is then to be adopted†:

Step 1. Check $y_i^s(i\epsilon M)$.

1a. If $y_i^s\geqq 0(i\epsilon M)$, set $z_s=z^{*(s)}$. Form the sets D_k^s as defined by (17) for all $k<s$, cancel all v_j^k for $j\epsilon D_k^s$, $k<s$, and pass to step 5.

If this happens to be the case for u^0, then u^0 is an optimal solution and the algorithm has ended.

1b. If there exists i_1 such that $y_{i_1}^s<0$, pass to

† The algorithm can be followed on the flow chart presented in Fig. 2.

Step 2. Identify the improving vectors for the solution u^s by forming the set N_s as defined by (16).

2a. If $N_s=\phi$, i.e., there are no improving vectors for u^s, pass to step 5.

2b. If $N_s\neq\phi$, pass to

Step 3. Check the relations

$$\sum_{j\in N_s} a_{ij}\leqq y_i^s, \qquad (i|y_i^s<0) \quad (22)$$

where a_{ij} are the negative elements of A.

3a. If there exists $i_1\in M$ for which (22) does not hold, pass to step 5.

3b. If all relations (22) hold as strict inequalities, compute the values v_j^s as defined by (11) and (12) for all $j\in N_s$, choose j_{s+1} so that

$$v_{j_{s+1}}^s=\max_{j\in N_s} v_j^s,\ddagger \quad (23)$$

cancel $v_{j_{s+1}}^s$ and pass to step 8.

3c. If all relations (22) hold, and there exists a subset M^s of M such that the relations (22) hold as equalities for $i\in M^s$, pass to

Step 4. Check the relation

$$\sum_{j\in F_s} c_j<z^{*(s)}-z_s, \quad (24)$$

where F_s is the set of those $j\in N_s$, for which $a_{ij}<0$ for at least one $i\in M^s$.

4a. If (24) holds, cancel v_j^s for all $j\in F_s$ (without computing their numerical values). Set $J_{s+1}=J_s\cup F_s$, compute the value of the objective function

$$z_{s+1}=z_s+\sum_{j\in F_s} c_j \quad (25)$$

and of the slack variables

$$y_i^{s+1}=y_i^s-\sum_{j\in F_s} a_{ij} \qquad (i\in M) \quad (26)$$

for the new solution u^{s+1}, and pass to the next iteration (i.e., start again with step 1).

4b. If (24) does not hold, cancel v_j^s for all $j\in N_s$ (without computing their numerical values) and pass to

Step 5. Identify the improving vectors for the solutions u^k $(k|J_k\subset J_s)$ left after iteration s, i.e., check the sets N_k^s $(k|J_k\subset J_s)$ as defined by (18), in the decreasing order of the numbers k, until either a number k_1 is found such that $J_{k_1}\subset J_s$ and $N_{k_1}^s\neq\phi$, or N_k^s is found to be void for any k such $J_k\subset J_s$.

5a. If $N_k^s=\phi$ for all k such that $J_k\subset J_s$, i.e., there are no improving vectors for any u^k $(k|J_k\subset J_s)$, the algorithm has come to an end. In this case, if $Z_s=\phi$, P has no feasible solution. If $Z_s\neq\phi$, then u^q for which $z_q=z^{*(s)}$ is an optimal solution and $z^{*(s)}$ is the minimum attained by the objective function for this solution.

‡ If (23) or (28) holds for more than one $j=j_{s+1}$ (let $J_{\max}$ be the set of those j for which it holds), choose j_{s+1} so that $c_{j_{s+1}}=\min_{j\in J_{\max}} c_j$, and if this relation too holds for more than one $j=j_{s+1}$, then choose any one of them as j_{s+1}.

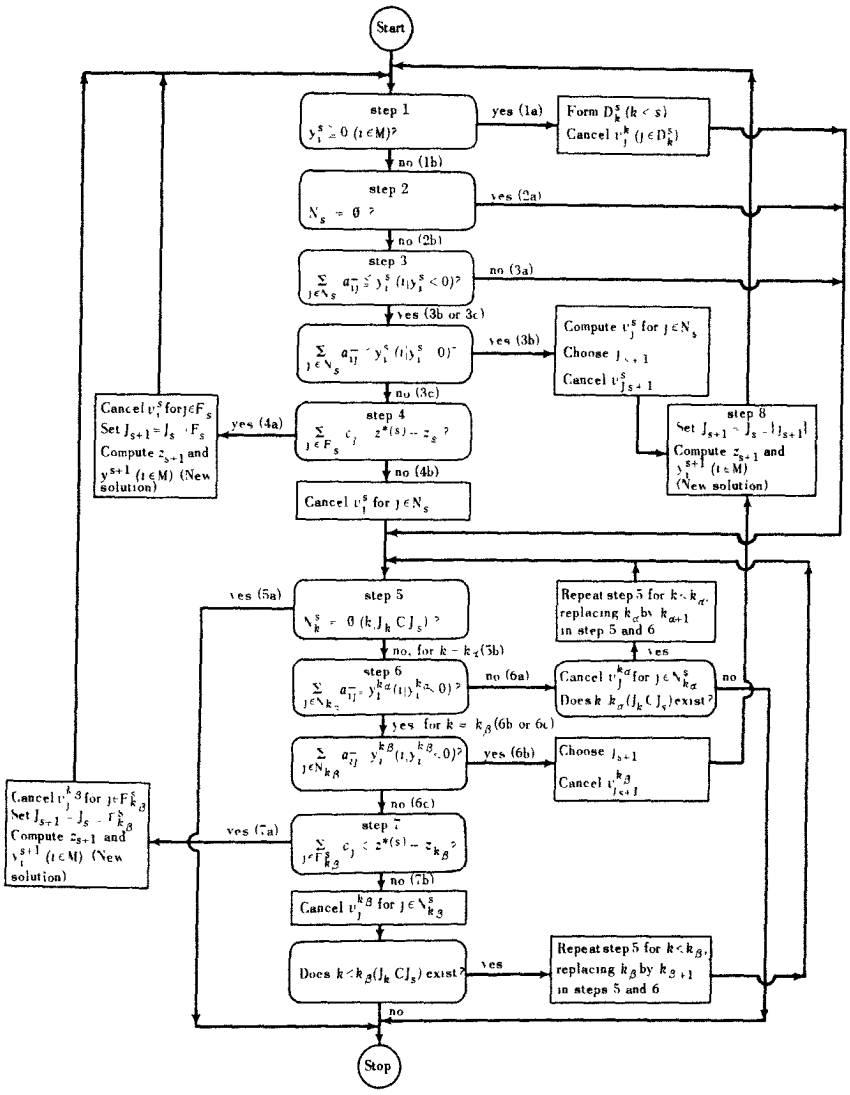

Fig. 2. Flow chart of the additive algorithm.

5b. If $N_k^s \neq \phi$ for $k=k_1, J_{k_1} \subset J_s$, pass to

Step 6. Check the relations

$$\sum_{j \in N_k^s} a_{ij}^- \leqq y_i^k \qquad (i|y_i^k<0) \quad (27)$$

for $k=k_1$.

6a. If any one of the relations (27) does not hold for $k=k_1$, cancel $v_j^{k_1}$ for all $j \in N_{k_1}^s$ and repeat step 5 for $k<k_1$, writing k_2 instead of k_1 in steps

5 and 6. Whenever step 5 is repeated for $k<k_\alpha$, write $k_{\alpha+1}$ instead of k_α in steps 5 and 6.

If (27) does not hold for any k such that $N_k^s \neq \phi$, the algorithm has ended, with the same conclusion as for 5a.

6b. If for $k=k_\beta$ all relations (27) hold as strict inequalities, choose j_{s+1} so that

$$v_{j_{s+1}}^{k_\beta} = \max_{j \in N_{k_\beta}^s} v_j^{k_\beta}, \tag{28}$$

cancel $v_{j_{s+1}}^{k_\beta}$ and pass to step 8.

6c. If for $k=k_\beta$ all relations (27) hold, and there exists a subset $M_{k_\beta}^s$ of M such that the relations (27) hold as equalities for $i \in M_{k_\beta}^s$, pass to

Step 7. Check the relation

$$\sum_{j \in F_{k_\beta}^s} c_j < z^{*(s)} - z_{k_\beta}, \tag{29}$$

where $F_{k_\beta}^s$ is the set of those $j \in N_{k_\beta}^s$, for which $a_{ij}<0$ for at least one $i \in M_{k_\beta}^s$.

7a. If (29) holds, cancel $v_j^{k_\beta}$ for all $j \in F_{k_\beta}^s$. Set $J_{s+1}=J_{k_\beta} \cup F_{k_\beta}^s$, compute the value of the objective function

$$z_{s+1} = z_{k_\beta} + \sum_{j \in F_{k_\beta}^s} c_j \tag{30}$$

and of the slack variables

$$y_i^{s+1} = y_i^{k_\beta} - \sum_{j \in F_{k_\beta}^s} a_{ij} \qquad (i \in M) \tag{31}$$

for the new solution $\boldsymbol{u}^{s+1}$ and pass to the next iteration (i.e., start again with step 1).

7b. If (29) does not hold, cancel the values $v_j^{k_\beta}$ for all $j \in N_{k_\beta}^s$ and repeat step 5, for $k<k_\beta$. If no $k<k_\beta$ exists, i.e., if $k_\beta=0$, the algorithm has ended with the same conclusion as for 5a.

Step 8. Set $J_{s+1}=J_p \cup \{j_{s+1}\}$ and compute the value of the objective function and of the slack variables for the new solution $\boldsymbol{u}^{s+1}$, according to the formulas

$$z_{s+1} = z_p + c_{j_{s+1}} = \sum_{j \in J_{s+1}} c_j \tag{32}$$

and

$$y_i^{s+1} = y_i^p - a_{ij_{s+1}} = b_i - \sum_{j \in J_{s+1}} a_{ij}, \qquad (i \in M) \tag{33}$$

where p is defined by the last value cancelled, $v_{j_{s+1}}^p$. Pass to the next iteration (i.e., start again with step 1).

The algorithm terminates when a solution $\boldsymbol{u}^t$ has been reached, for which (α) situation 5a obtains; or (β) situation 6a obtains and (27) does not hold for any k such that $N_k^s \neq \phi$; or (γ) situation 7b obtains and $k_\beta=0$.

REMARK I. The additive algorithm described above yields *one* optimal solution (if such a solution exists). But if we set $>$ instead of $\geqq$ in (14) and (17), and $\leqq$ instead of $<$ in (24) and (29), we obtain another version of the same algorithm yielding *all* the existing optimal solutions.

REMARK II. On the other hand, the number of solutions to be examined

under the algorithm may be further reduced at the cost of a relatively small additional computational effort at each iteration, if we change steps 3b, 3c and 6b, 6c as follows:

3b. If all relations (22) hold, check the relations

$$\sum_{j\epsilon N_s} a_{\bar{i}j} - a_{\bar{i}j_s} \leqq y_i^s \qquad (i|y_i^s<0)- \quad (22a)$$

where $a_{\bar{i}j}=\max_{j\epsilon N_s} a_{\bar{i}j}$; if all relations (22a) hold, compute the values v_j^s for all $j\epsilon N_s$, choose j_{s+1} so that

$$v_{j_{s+1}}^s=\max_{j\epsilon N_s} v_j^s; \qquad (23)$$

cancel $v_{j_{s+1}}^s$ and pass to step 8.

3c. If all relations (22) hold, and there exists a subset M^s of M such that relations (22a) do not hold for $i\epsilon M^s$, pass to step 4.

Completely analogous changes are to be made in steps 6b, 6c.

The application of the algorithm is facilitated through the use of a tableau of the type presented with the numerical examples of the final section.

FINITENESS PROOFS

THE ADDITIVE algorithm yields a sequence of solutions $u^0, u^1, \cdots$. We shall say that a solution u^k is *abandoned*, if we are instructed by the algorithm either to check $N_p^*(0\leqq p<k\leqq s)$, or to stop (termination).

A central feature of our algorithm, which is at the same time instrumental in proving its convergence, may be expressed by Theorem 1.

THEOREM 1. *If a solution u^k is abandoned under the additive algorithm, then no feasible solution u^t exists such that $J_k\subset J_t$ and $z_t<z^{*(k)}$.*

A solution may be abandoned as a consequence of one of the following steps of the algorithm: 1a, 2a, 3a, 4b, 5, 6a, and 7b. In order to prove that the above theorem holds for all these situations, we shall need two lemmas.

First, let us consider the sets C_p^s. Under the rules of our algorithm, there are three circumstances in which a value $v_{j_*}^p$ assigned to a solution u^p may have been cancelled until the iteration s, i.e., there are three reasons for j_* to belong to a set C_p^s:

(a) Relation (27) does not hold for $k=p$ (i.e., j_* has been cancelled under step 6a).

(b) A solution $u^q(p<q\leqq s)$ has been obtained from u^p by introducing a_{j_*} into the basis (i.e., j_* has been cancelled under one of the steps 3b, 4a, 6b, 7a).

(c) If a_{j_*} were to be introduced into the basis, z would hit the ceiling for u^s (i.e., j_* has been cancelled under one of the steps 1a, 4b, or 7b).

Let us denote the subsets of indices j corresponding to those v_j^p that

have been cancelled until iteration s for reasons a, b, and c respectively, by $C_p^s(a)$, $C_p^s(b)$, and $C_p^s(c)$, so that $C_p^s = C_p^s(a) \cup C_p^s(b) \cup C_p^s(c)$, and let us denote by $C^s(a)$, $C^s(b)$, and $C^s(c)$ so that

$$C_p^s = C_p^s(a) \cup C_p^s(b) \cup C_p^s(c),$$

and let us denote by $C^s(a)$, $C^s(b)$, and $C^s(c)$ respectively, the corresponding subsets of C^s.

We have

LEMMA 1. *Given a solution* $\boldsymbol{u}^s$, *if there exists a feasible solution* $\boldsymbol{u}^t$ *such that* $J_s \subset J_t$ *and* $z_t < z^{*(s)}$, *then*

$$(J_t - J_s) \cap C^s = \phi. \tag{34}$$

Proof. Relation (34) is obviously satisfied for any solution $\boldsymbol{u}^s$ such that $C^s = J_s$ and this holds in any case for $\boldsymbol{u}^0$, as $J_0 = C^0 = \phi$.

Thus we are entitled to suppose that (34) holds for a sequence of solutions $\boldsymbol{u}^p$ $(p = 0, 1, \cdots)$, composed of at least one term.

We shall prove that if (34) holds for $\boldsymbol{u}^p$ $(p = 0, 1, \cdots, n)$, then it also holds for $\boldsymbol{u}^{n+1}$.

Let us denote $\boldsymbol{u}^n = \boldsymbol{u}(j_1, \cdots, j_r)$. Two situations are then possible:

(1) $J_n \subset J_{n+1}$.

In this case either $J_{n+1} = J_n \cup \{j_{n+1}\}$ and $C^{n+1} = C^n \cup \{j_{n+1}\}$, or $J_{n+1} = J_n \cup F_n$, and $C^{n+1} = C^n \cup F_n$. In both cases (34) obviously holds for $\boldsymbol{u}^{n+1}$.

(2) $J_n \not\subset J_{n+1}$.

Let us denote $J_k = \{j_1, \cdots, j_{r-h}\}$, h being the smallest number $(1 \leqq h \leqq r)$ for which $J_k \subset J_{n+1}$.

According to the definition of C^{n+1},

$$C^{n+1} = \bigcup_{p|J_p \subseteq J_k} C_p^{n+1} = [\bigcup_{p|J_p \subseteq J_k} (C_p^{n+1} - C_p^k)] \cup C^k. \tag{35}$$

As for $\boldsymbol{u}^k$ (34) is supposed to hold, and as $J_k \subset J_{n+1}$ and $z^{*(k)} \geqq z^{*(n+1)}$, we have,

$$(J_t - J_{n+1}) \cap C^k = \phi \tag{36}$$

for any $\boldsymbol{u}^t \geqq \boldsymbol{0}$ such that $J_{n+1} \subset J_t$ and $z_t < z^{*(n+1)}$.

On the other hand, as $J_k \subset J_{n+1}$, we have $C_p^{n+1}(a) = \phi (p|J_p \subseteq J_k)$, and $C_p^{n+1}(b) = C_p^k(b) (p|J_p \subset J_k)$, so that

$$C_p^{n+1} - C_p^k = C_p^{n+1}(c) - C_p^k(c) (p|J_p \subset J_p \subset J_k), \tag{37}$$

and

$$C_k^{n+1} - C_k^k = C_k^{n+1} = C_k^{n+1}(c) \cup C_k^{n+1}(b). \tag{38}$$

Further, by the definition of $C_p^{n+1}(c)$,

$$(J_t - J_{n+1}) \cap \{\bigcup_{p|J_p \subseteq J_k} [C_p^{n+1}(c) - C_p^k(c)]\} = \phi. \tag{39}$$

Thus, from the above relations it follows that

$$(J_t - J_{n+1}) \cap C^{n+1} = (J_t - J_{n+1}) \cap C_k^{n+1}(b). \tag{40}$$

Now, there are two possibilities: either $J_{n+1}=J_k \cup \{j_{n+1}\}$, or $J_{n+1}=J_k \cup F_k^n$.

Let us suppose that the first of these two situations holds, and thus $C_k^{n+1}(b)=C_k^n(b)\cup\{j_{n+1}\}$. (A perfectly analogous reasoning is valid for the second situation.)

We have

$$(J_t-J_{n+1})\cap C^{n+1}=(J_t-J_k)\cap(J_t-\{j_{n+1}\})\cap[C_k^n(b)\cup\{j_{n+1}\}], \tag{41}$$

or, as

$$(J_t-\{j_{n+1}\})\cap j_{n+1}\}=\phi, \tag{42}$$

$$(J_t-J_{n+1})\cap C^{n+1}=(J_t-J_k)\cap(J_t-\{j_{n+1}\})\cap C_k^n(b). \tag{43}$$

We shall now show that

$$(J_t-J_k)\cap C_k^n(b)=\phi. \tag{44}$$

Let us suppose the contrary, i.e., that there exists j_{f_1} such that

$$j_{f_1}\epsilon[(J_t-J_k)\cap C_k^n(b)], \tag{45}$$

and let us denote $J_{q_1}=J_k\cup\{j_{f_1}\}$.

Obviously, J_t is not identical with J_{q_1}, as from $j_{f_1}\epsilon C_k^n(b)$ it follows that $q_1<n$, i.e., $z^{*(q_1)}\geqq z^{*(n)}$, while $z^{*(t)}<z^{*(n)}$. Thus, if (45) holds, there must exist j_{f_2} such that

$$j_{f_2}\epsilon[(J_t-J_{q_1})\cap C_{q_1}^n(b)]. \tag{46}$$

For, if not, then

$$(J_t-J_{q_1})\subseteq[N-C_{q_1}^n(b)]=C^{q_1}\cup D_{q_1}\cup E_{q_1}\cup[N_{q_1}-C_{q_1}^n(b)] \tag{47}$$

must hold. But, as $k<q_1<n$, and as $j_{f_1}\epsilon C_k^n(b)$ implies $J_{q_1}\not\subset J_{n+1}$, we have $N_{q_1}=C_{q_1}^{n+1}$, and $C_{q_1}^{n+1}(b)=C_{q_1}^n(b)$. Thus

$$N_{q_1}-C_{q_1}^n(b)=C_{q_1}^{n+1}(a)\cup C_{q_1}^{n+1}(c). \tag{48}$$

On the other hand, from the definition of D_{q_1}, $C_{q_1}^{n+1}(a)$ and $C_{q_1}^{n+1}(c)$, and from the fact that (34) is supposed to hold for q_1 as $q_1<n$ and $J_{q_1}\subset J_n$, it follows that

$$(J_t-J_{q_1})\cap[C^{q_1}\cup D_{q_1}\cup C_{q_1}^{n+1}(a)\cup C_{q_1}^{n+1}(c)]=\phi, \tag{49}$$

and, by the definition of E_{q_1},

$$J_t-J_{q_1}\not\subset E_{q_1}. \tag{50}$$

Therefore (47) is impossible, i.e., (46) must hold if (45) holds.

Now, if $C_{q_1}^n(b)=\phi$, we have a contradiction and the validity of (44) is proved. If not, then a reasoning perfectly analogous to the above one shows that, if

$$j_{f_i}\epsilon[(J_t-J_{q_{i-1}})\cap C_{q_{i-1}}^n(b)] \tag{51}$$

exists, then

$$j_{f_{i+1}}\epsilon[(J_t-J_{q_i})\cap C_{q_i}^n(b)] \tag{52}$$

must also exist ($i=2, 3, \cdots$), $J_{q_{i-1}}$ and J_{q_i} being defined analogously to J_{q_1}. As the number of sets $C^n_{q_i}(b) \neq \phi$ is finite, this sequence of implications obviously ends in a contradiction that proves the validity of (44).

Relation (34) is thus shown to hold also in case (2), and this completes the proof of Lemma 1.

LEMMA 2. *Given two solutions* $\boldsymbol{u}^s = \boldsymbol{u}(j_1, \cdots, j_r)$ *and* $\boldsymbol{u}^k = \boldsymbol{u}(j_1, \cdots, j_{r-h})$, $(1 \leqq h \leqq r;\ J_k \subset J_s)$, *such that* $N_p = C_p^{s+1} (k < p \leqq s)$, *if there exists a feasible solution* $\boldsymbol{u}^t$ *such that* $J_k \subset J_t$ *and* $z_t < z^{*(s)}$, *then*

$$(J_t - J_k) \cap C_k^s = \phi. \tag{53}$$

Proof. $J_k \subset J_s$ guarantees that $C_k^s(a) = \phi$, while

$$(J_t - J_k) \cap C_k^s(c) = \phi \tag{54}$$

follows from the definition of $C_k^s(c)$. Thus (53) becomes

$$(J_t - J_k) \cap C_k^s(b) = \phi, \tag{55}$$

which can be proved in exactly the same way as (44).

Proof of Theorem 1. For solutions abandoned under one of the steps 1a, 4b, or 7b, the theorem is obvious. We shall now prove it for the other cases:

2a. In this situation, as $N_k = \phi$, the existence of a feasible solution $\boldsymbol{u}^t$ such that $J_k \subset J_t$, $z_t < z^{*(k)}$, would imply

$$J_t - J_k \subseteq N - N_k = C^k \cup D_k \cup E_k, \tag{56}$$

which is impossible in view of lemma 2 and the definitions of D_k and E_k.

3a. If (22) does not hold, then, in view of the definition of E_k, a feasible solution $\boldsymbol{u}^t$ with the required properties could only exist if

$$(J_t - J_k) \cap (C^k \cup D_k) \neq \phi, \tag{57}$$

which cannot hold according to Lemma 1 and by the definition of D_k.

5. If $N_k^s = \phi$ for some k, the existence of $\boldsymbol{u}^t$ with the required properties would imply

$$J_t - J_k \subseteq N - N_k^s. \tag{58}$$

But in view of the definitions of D_k, D_k^s, and $C_k^s(c)$, no element of any of these sets can be contained in $J_t - J_k$. Further, $C_k^s(a) = \phi$ because the checking of N_k^s for a certain k under step 5 always precedes the possible cancellation of values v_j^k for the same k under step 6a.

On the other hand, the checking of N_k^s for a certain k under step 5 can never occur before $N_p = C_p^{s+1}$ has been established for all p such that $k < p \leqq s$. Therefore, according to Lemma 2, $(J_t - J_k) \cap C_k^s$ is also void and (58) becomes

$$J_t - J_k \subseteq E_k, \tag{59}$$

which is obviously impossible.

6a. If (27) does not hold for a certain k, then a solution u^t such as required in the theorem could only exist if

$$(J_t - J_k) \cap [N - (N_k^s \cup E_k)] \neq \phi. \tag{60}$$

But this cannot hold, as was shown at the preceding point.

Theorem 1 is thus proved.

Now we can formulate the following

CONVERGENCE THEOREM 2. *In a finite number of iterations, the additive algorithm yields either an optimal feasible solution, or the conclusion that the problem has no feasible solution at all.*

Proof. From theorem 1 it follows that the termination of the algorithm (which means that all solutions tested by the algorithm have been abandoned) yields either an optimal feasible solution, or the conclusion that no feasible solution to the problem exists. We shall now show that the additive algorithm terminates in a finite number of steps.

(a) In a finite number of steps, each iteration yields a new solution or else the algorithm terminates. Repetition of certain steps during one and the same iteration may arise in one of the situations 6a or 7b. In both situations step 5, which requires the checking of N_k^s $(k|J_k \subset J_s)$ is to be repeated for $k < k_\alpha$, where k_α is a number for which the checking had already taken place. Thus there can only be a finite number of such repetitions.

(b) No solution can be obtained twice under the additive algorithm. Let us suppose the contrary, i.e., let $u^s = u(j_1, \cdots, j_r)$ and $u^t = u(k_1, \cdots, k_r)$ be two solutions obtained under the additive algorithm, such that $u^s = u^t$, $t > s$. We cannot have $j_i = k_i (i = 1, \cdots, r)$, because in that case $j_r \epsilon C_q^{t-1}$ (where q is defined either by $J_t = J_q \cup \{j_r\}$ or by $J_t = J_q \cup F_q^{t-1}$), and this excludes $j_r \epsilon J_t$. Let (j_α, k_α) be the first pair of indices (j_i, k_i) such that $j_i \neq k_i$ and let us denote $u^p = u(j_1, \cdots, j_{\alpha-1})$. Then $j_\alpha \epsilon C_p^{t-1} \subset C^{t-1}$ and thus $j_\alpha \notin J_t$.

This completes the proof of the Convergence Theorem.

SOME REMARKS ON THE EFFICIENCY OF THE ALGORITHM

UNLIKE MOST of the known algorithms for solving linear programs with integer variables, the additive algorithm attacks *directly* the linear program with zero-one variables, and does not require the solution of the corresponding ordinary linear program (without the zero-one constraints).

As has been shown, the only operations required under the algorithm described above are additions and subtractions. Thus any possibility of round-off errors is excluded.

The additive algorithm does not impose a heavy burden on the storage system of a computer. Most of the partial results may be dropped shortly after they have been obtained.

At this moment, experience with the additive algorithm is at its very

beginning. So far it consists only of solving by hand about a dozen problems with up to 15 variables, and of partially solving, also by hand, one single large problem. The results are very encouraging, but of course this experience is insufficient for a firm judgment on the efficiency of the algorithm, especially for larger problems. We shall summarize the above experience and comment on it, but all our conjectures in this section should be regarded as tentative, in view of their scanty experimental basis.

The data of 11 problems solved by hand with the additive algorithm are given in Tableau I.

TABLEAU I

Problem no.	Number of zero-one variables	Number of constraints (inequalities)	Number of iterations
1	5	2	6
2[a]	5	3	3
3	7	5	11
4[a][b]	9	4	31
5	10	4	12
6[a]	10	7	5
7	11	6	21
8	11	7	8
9[a][c]	12	6	39
10	14	9	23
11	15	12	22

[a] Problems given as numerical examples in final section.
[b] A problem with no feasible solution.
[c] A problem with only one feasible solution.

The time needed for solving these problems† was on the average 7 minutes for an iteration, i.e., 1–3 hours for each of the problems 3, 4, 5, 7, 8, 10, and 11. Problems 1, 2, and 6 were solved in less than an hour, while somewhat more than 4 hours were used to solve problem 9.

Any one who has tried to solve by hand a problem of the type and size discussed here with the cutting-plane technique for integer programming problems (which in this case has to be combined with bounded variables-procedures) knows that it takes several times that amount of time, not to speak about difficulties generated by round-off. Moreover, the solution of the ordinary linear programs (without the zero-one constraints) corresponding to the above problems would also require an amount of computations considerably larger than that needed for solving the zero-one problems by the additive algorithm.

† By hand calculations by a person having no special experience with the additive algorithm.

The only large problem on which the additive algorithm has so far been tried was a problem in forest management,[10] with a linear objective function in zero-one variables subject to a set of linear constraints, and to another set of conditional ('logical') constraints.‡ The latter set has been replaced by an equivalent set of linear constraints including additional zero-one variables, so that finally a zero-one linear programming problem emerged with 40 variables and 22 constraints. The number of nonzero elements in the final coefficient matrix was 140, of which 115 were 1 or -1. This problem was specially chosen for studying the applicability of the algorithm to the given type of forest-management problems, and it was so structured that the optimal solution could be known beforehand. After 35 iterations were made by hand—the average time of an iteration being about 20 minutes—the optimal feasible solution was approximated within 1.4 per cent. We note, however, that this approximation cannot be regarded as a sign that termination of our algorithm was also near, and we further note that our coefficient-matrix was relatively sparse—with many 0 elements.

This experiment showed, among other things, the great advantages of the additive character of the algorithm. For instance, an error discovered at the 19th iteration could be traced back to the 3rd iteration and corrected throughout the subsequent solutions in about 2 hours' time.

The following considerations concerning the dependence of the amount of computations on the size and other characteristics of the problem are based partly on common-sense examination of the mechanism of the algorithm, partly on the experience summarized above.

Let n be the number of variables and m the number of constraints.

(a) *Amount of computations needed for one iteration.* The number of operations needed for checking relations (22) and (27) (steps 3 and 6 respectively), and for computing the values v_j^s (step 3b), depends linearly on $m \times n$. The number of operations needed to form or check the sets N_s and N_k^s (steps 2, 5), depends linearly on n, while the number of operations needed to compute and check a new solution (steps 8, 4a, 7a, and 1a) depends linearly on m. Thus the total amount of computations needed for one iteration depends linearly on a quantity μ situated somewhere between $\min[m, n]$ and $m \times n$.

(b) *Number of iterations needed to solve the problem.* This obviously depends first of all on n. As to m, its increase enhances the efficiency of some of the stop signals (steps 3a and 6a) and thus tends to reduce the number of iterations. (This is an important advantage of the algorithm.)

The crucial thing about the efficiency of the algorithm is to know how the number of iterations depends on n. The combinatorial nature of the algorithm does not necessarily imply an exponential relation for, while the

‡ Expressing relations of the type $\vee$, $\Rightarrow$, and $\Leftrightarrow$.

set of all solutions from which an optimal feasible one has to be selected grows exponentially with n, the efficiency of some of the stop signals may perhaps grow even faster with n. So far the experience summarized above does not seem to indicate an exponential relation. While in the three smallest problems of Tableau I (1, 2, and 3) the number of iterations was 0.6–1.6 times n, in the two largest problems (10 and 11) it was 1.5–1.6 times n. In the largest problem so far solved (11), 22 solutions out of a total of $2^{15}=32{,}768$ had to be tested. But of course this experience is insufficient, and computer experience with a considerable number of larger problems will be needed to elucidate this question in the absence of an analytic proof.

The number of iterations also depends on other characteristics of the problem. In cases where an optimal solution exists in which only a few variables take the value 1, the relatively large size of the sets D_s makes the stop signals especially efficient and thus assures a rapid convergence of the algorithm (*see*, for instance, problem 6 of Tableau I) (example 2 in the final section), where only 5 out of $2^{10}=1{,}024$ solutions had to be tested). On the other hand, if the problem has no feasible solution, all sets D_s are void and the efficiency of the stop signals is reduced. The same holds to a lesser extent for problems with very few feasible solutions. But it should be noted that even in the case of such 'ill-behaved' problems the number of iterations does not become unreasonably large. Thus, problem 4 of Tableau I (example 3 in the final section), with 9 variables and 4 constraints, having no feasible solution, was 'solved' (i.e., the absence of a feasible solution was established) in 31 iterations, while in the worst of cases met until now, problem 9 of Tableau I (example 4 in the final section), with 12 variables and 6 constraints, having only one feasible solution, was solved by testing 39 out of $2^{12}=4{,}096$ solutions.

NUMERICAL EXAMPLES

Example 1. (Problem 2 of Tableau I.)
Let us consider the following problem:

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

Multiplying by -1 the two inequalities of the form $\geqq$ and setting

$$x_j=\begin{cases} x_j', & (j=2, 3, 5) \\ 1-x_j', & (j=1, 4) \end{cases}$$

allows us to restate our problem in the following form, corresponding to P in the second section:

$$\begin{aligned} 5x_1+7x_2+10x_3+3x_4+\ x_5 &= \min, \\ -x_1+3x_2-\ 5x_3-\ x_4+4x_5+y_1 &= -2, \\ 2x_1-6x_2+\ 3x_3+2x_4-2x_5\quad +y_2 &= 0, \\ x_2-\ 2x_3+\ x_4+\ x_5\quad +y_3 &= -1, \\ x_j=0 \text{ or } 1. \qquad (j=1, \cdots, 5) \end{aligned}$$

TABLEAU II

Row no.	s	J_s	z_s	N_s		i			v_j^s	C^s	D_s	E_s	F_s
						1	2	3					
1	0	ϕ	0		y_i^0	−2	0	−1		ϕ	ϕ	2, 5	
2					$\sum_{j\in N_0} a_{ij}^-$	−7		−2					ϕ
3				1	$y_i^0-a_{i1}$	−1	−2	−1	-4^4				
4				3	$y_i^0-a_{i3}$		−3		-3^1				
5				4	$y_i^0-a_{i4}$	−1	−2	−2	-5^5				
6	1	3	10		y_i^1	3	−3	1		3	ϕ	1, 4	
7					$\sum_{j\in N_1} a_{ij}^-$		−2						ϕ
8				2	$y_i^1-a_{i2}$				0^2				
9				5	$y_i^1-a_{i5}$	−1	−1		-2^3				
10	2	3, 2	17*		y_i^2	0	3	0					
11					$\sum_{j\in N_1^2} a_{ij}^-$		−2						ϕ
12					$\sum_{j\in N_0^2} a_{ij}^-$	−2		0					

In view of the form of Tableau II, the computations are easier to follow if we work with A^T instead of A (T being the symbol of transposition):

$$A^T = \begin{Vmatrix} -1 & 2 & 0 \\ 3 & -6 & 1 \\ -5 & 3 & -2 \\ -1 & 2 & 1 \\ 4 & -2 & 1 \end{Vmatrix}.$$

We start with the initial solution $u^0=(\mathbf{0}, \boldsymbol{b})$. The data of this solution, $J_0=\phi$, $z_0=0$, and $y_1^0=b_1=-2$, $y_2^0=b_2=0$, $y_3^0=b_3=-1$, are shown on the left-hand side of row 1 of Tableau II.

To make this illustration easier to follow on Tableau II, cancellation of the values v_j^s is not marked through crossing out, but through numbering

in the order of cancellation. (Of course, this is not necessary in current work with the algorithm.)

The solution of this problem is also illustrated by Fig. 1.

Iteration 1

Step 1. $y_i^0 < 0$ for $i = 1, 3$; so we are in situation 1b.

Step 2. $N_0 = N - (C^0 \cup D_0 \cup E_0)$. $C^0 = \phi$, $D_0 = \phi$, $E_0 = \{2, 5\}$ (*see* row 1 of the Tableau).

$N_0 = \{1, 2, 3, 4, 5\} - \{2, 5\} = \{1, 3, 4\}$; so we are in case 2b.

Step 3. We check the relations (22) for $i = 1, 3$ (*see* row 2 of the Tableau).

$$\sum_{j \in N_0} \bar{a}_{1j} = \bar{a}_{11} + \bar{a}_{13} + \bar{a}_{14} = -1 - 5 - 1 = -7 < y_1^0 = -2,$$
$$\sum_{j \in N_0} \bar{a}_{3j} = \bar{a}_{33} = -2 < y_3^0 = -1.$$

As all relations hold as strict inequalities, we are in case 3b.

Computation of the values v_j^0 as defined by (11), (12), for $j \in N_0$, i.e., for $i = 1, 3, 4$, is shown in rows 3, 4, 5 of the Tableau:

$$v_1^0 = \sum_{i \in M_1^0} (y_i^0 - a_{i1}) = (-2+1) + (0-2) + (-1-0) = -4,$$
$$v_3^0 = \sum_{i \in M_3^0} (y_i^0 - a_{i3}) = (0-3) = -3,$$
$$v_4^0 = \sum_{i \in M_4^0} (y_i^0 - a_{i4}) = (-2+1) + (0-2) + (-1-1) = -5.$$

We have $v_{j_1}^0 = \max_{j \in N_0} \{v_j^0\} = v_3^0 = -3$, we cancel v_3^0 and pass to

Step 8. $J_1 = J_0 \cup \{3\} = \{3\}$,

$$z_1 = z_0 + c_3 = 0 + 10 = 10,$$
$$y_1^1 = y_1^0 - a_{13} = -2 + 5 = 3,$$
$$y_2^1 = y_2^0 - a_{23} = 0 - 3 = -3,$$
$$y_3^1 = y_3^0 - a_{33} = -1 + 2 = 1,$$

(*see* row 6 of Tableau II).

Iteration 2

Step 1. $y_i^1 < 0$ for $i = 2$, so we are in situation 1b.

Step 2. $N_1 = N - (C^1 \cup D_1 \cup E_1)$. $C_1 = \{3\}$, $D_1 = \phi$, $E_1 = \{1, 4\}$ (*see* row 6 of the Tableau).

$N_1 = \{1, 2, 3, 4, 5\} - (\{3\} \cup \{1, 4\}) = \{2, 5\}$, so we are in case 2b.

Step 3. Checking of the relations (22) for $i = 2$ is shown in row 7 of the Tableau:

$$\sum_{j \in N_1} \bar{a}_{2j} = -6 - 2 = -8 < y_2^1 = -2.$$

Thus we are in case 3b. Computation of v_j^1 for $j \in N_1$ is shown in rows 8 and 9 of Tableau II.

$$v_2^1 = 0 (M_2^{1-} = \phi),$$
$$v_5^1 = \sum_{j \in M_5^{1-}} (y_i^1 - a_{i5}) = (3-4) + (-3+2) + (1-1) = -2.$$

We have $v_{j_1}^1 = \max_{j \in N_1}\{v_j^1\} = v_2^1 = 0$, we cancel v_2^1 and pass to

Step 8. $J_2 = J_1 \cup \{2\} = \{3, 2\}$,

$z_2 = z_1 + c_2 = 10 + 7 = 17$,

$y_1^2 = y_1^1 - a_{12} = 3 - 3 = 0$,

$y_2^2 = y_2^1 - a_{22} = -3 + 6 = 3$,

$y_3^2 = y_3^1 - a_{32} = 1 - 1 = 0$,

(*see* row 10 of Tableau II).

Iteration 3

Step 1. $y_i^2 \geqq 0$ for all $i \in M$, so we are in case 1a. We set $z_2 = 17 = z^{*(2)}$, and we form the sets D_k^2 as defined by (17) for $k = 1, 0$:

$N_1 = \{2, 5\}$, $C_1^2 = \{2\}$; $N_1 - C_1^2 = \{5\}$; $c_5 = 1 < 17 - 10$, so $D_1^2 = \phi$,

$N_0 = \{1, 3, 4\}$, $C_0^2 = \{3\}$; $N_0 - C_0^2 = \{1, 4\}$; $c_1 = 5 < 17 - 0$, $c_4 = 3 < 17 - 0$;

$D_0^2 = \phi$.

We pass to

Step 5. We check the set N_k^2 as defined by (18) for $k = 1$:

$N_1^2 = N_1 - (C_1^2 \cup D_1^2) = \{2, 5\} - \{2\} = \{5\}$, so we are in situation 5b.

Step 6. We check the relations (27) for $i = 2$ (row 11 of Tableau II):

$$\sum_{j \in N_1^2} \bar{a_{2j}} = -2 \nleqq y_2^2 = -3.$$

As (27) does not hold, we are in case 6a. We cancel v_j^1 for $j \in N_1^2$, i.e., v_5^1, and return to

Step 5. We check the set N_k^2 for $k = 0$:

$N_0^2 = N_0 - (C_0^2 \cup D_0^2) = \{1, 3, 4\} - \{3\} = \{1, 4\}$ so we are again in case 5b.

Step 6. We check (27) for $i = 1, 3$ (row 12 of the Tableau):

$$\sum_{j \in N_0^2} \bar{a_{1j}} = -1 - 1 = -2 = y_1^0 = -2,$$

$$\sum_{j \in N_0^2} \bar{a_{3j}} = 0 \nleqq -1.$$

As (27) does not hold for $i = 3$, we are again in case 6a, and we cancel v_j^0 for $j \in N_0^2$, i.e., v_1^0 and v_4^0.

As (27) does not hold for any k such that $N_k^2 \neq \phi$, the algorithm has terminated. The optimal solution obtained for the restated problem is $\boldsymbol{u}^2 = \boldsymbol{u}(3, 2)$, with

$$x_2^2 = x_3^2 = 1, \qquad x_1^2 = x_4^2 = x_5^2 = 0, \qquad y_1^2 = 0, \qquad y_2^2 = 3, \qquad y_3^2 = 0,$$

and $z_2 = 17$.

The corresponding optimal solution to the initial problem is

$$x_1' = x_2' = x_3' = x_4' = 1, \qquad x_5' = 0,$$

the value of the objective function being $\boldsymbol{c}'\boldsymbol{x}' = 9$.

Example 2. (Problem 6 of Tableau I).

We shall consider the following problem:

$$
\begin{array}{rrrrrrrrrrrl}
10x_1' & -7x_2' & +x_3' & -12x_4' & +2x_5' & +8x_6' & -3x_7' & -x_8' & +5x_9' & +3x_{10}' & = \max, \\
3x_1' & +12x_2' & -8x_3' & -x_4' & & & & & -7x_9' & +2x_{10}' & \geqq -8, \\
 & x_2' & +10x_3' & & +5x_5' & -x_6' & +7x_7' & +x_8' & & & \leqq 13, \\
-5x_1' & -3x_2' & +x_3' & & & & & -2x_8' & & -x_{10}' & = -6, \\
 & & -4x_3' & +2x_4' & & -5x_6' & -x_7' & +9x_8' & -2x_9' & & \geqq -8, \\
 & -9x_2' & & +12x_4' & -7x_5' & +6x_6' & & -2x_8' & -15x_9' & -3x_{10}' & \geqq -12, \\
8x_1' & +5x_2' & -2x_3' & -7x_4' & +x_5' & & -5x_7' & & +10x_9' & & \leqq 16, \\
 & & & & x_j' = 0 & \text{or} & 1. & & & & (j = 1, \cdots, 10)
\end{array}
$$

After replacing the equation in the above set through two inequalities, multiplying by -1 the objective function, and all inequalities of the form $\geqq$, and setting

$$
x_j = \begin{cases} x_j', & (j = 2, 4, 7, 8) \\ 1 - x_j', & (j = 1, 3, 5, 6, 9, 10) \end{cases}
$$

we obtain the restated form of the problem, corresponding to P in the second section, where

$$
\boldsymbol{c} = (c_1, \cdots, c_{10}) = (10, 7, 1, 12, 2, 8, 3, 1, 5, 3)
$$

$$
\mathbf{A}^T = \begin{Vmatrix}
3 & & 5 & -5 & & & -8 \\
-12 & 1 & -3 & 3 & & 9 & 5 \\
-8 & -10 & -1 & 1 & -4 & & 2 \\
1 & & & & -2 & -12 & -7 \\
 & -5 & & & & -7 & -1 \\
 & 1 & & & -5 & 6 & \\
 & 7 & & & 1 & & -5 \\
 & 1 & -2 & 2 & -9 & 2 & \\
-7 & & & & -2 & -15 & -10 \\
2 & & 1 & -1 & & -3 &
\end{Vmatrix}
\qquad
\mathbf{b} = \begin{Vmatrix} -2 \\ -1 \\ -1 \\ 1 \\ -3 \\ -7 \\ -1 \end{Vmatrix}.
$$

The application of the algorithm may be followed on Tableau III. Numbering of the v_j^s shows again the order of their cancellation.

The optimal solution is $\boldsymbol{u}^2 = \boldsymbol{u}\ (9, 3)$, with

$$
x_9^2 = x_3^2 = 1, \qquad x_1^2 = x_2^2 = x_4^2 = x_5^2 = x_6^2 = x_7^2 = x_8^2 = x_{10}^2 = 0;
$$

$y_1^2 = 13$, $y_2^2 = 9$, $y_3^2 = y_4^2 = 0$, $y_5^2 = 3$, $y_6^2 = 8$, $y_7^2 = 7$, and $z_2 = 6$.

The corresponding optimal solution to the initial problem is thus

$$
x_1' = x_5' = x_6' = x_{10}' = 1, \qquad x_2' = x_3' = x_4' = x_7' = x_8' = x_9' = 0;
$$

the value of the objective function being $\boldsymbol{c}'\boldsymbol{x}' = 23$.

TABLEAU III

Row no.	s	J_s	z_s	N_s		i							v_j^s	C_s	D_s	E_s	F_s
						1	2	3	4	5	6	7					
1	0	ϕ	0		y_i^0	−2	−1	−1	1	−3	−7	−1		ϕ	ϕ	ϕ	
2					$\Sigma_{j\in N_0}\bar{a}_{ij}$	−27	−15	−6		−22	−34	−31					ϕ
3				1	$y_i^0-a_{i1}$	−5	−1	−6		−3	−7		-22^3				
4				2	$y_i^0-a_{i2}$		−2		−2	−3	−16	−6	-29^4				
5				3	$y_i^0-a_{i3}$						−7	−3	-10^{15}				
6				4	$y_i^0-a_{i4}$	−3	−1	−1		−1			-6^5				
7				5	$y_i^0-a_{i5}$	−2		−1		−3			-6^{12}				
8				6	$y_i^0-a_{i6}$	−2	−2	−1			−13	−1	-19^8				
9				7	$y_i^0-a_{i7}$	−2	−8	−1		−4	−7		-22^{16}				
10				8	$y_i^0-a_{i8}$	−2	−2		−1		−9	−1	-15^{17}				
11				9	$y_i^0-a_{i9}$		−1	−1		−1			-3^1				
12				10	$y_i^0-a_{i10}$	−4	−1	−2		−3	−4	−1	-15^{18}				
13	1	9	5		y_i^1	5	−1	−1	1	−1	8	9		9	ϕ	1, 7, 10	
14					$\Sigma_{j\in N_1}\bar{a}_{ij}$		−15	−6		−22							ϕ
15				2	$y_i^1-a_{i2}$		−2		−2	−1	−1		-6^7				
16				3	$y_i^1-a_{i3}$								0^3				
17				4	$y_i^1-a_{i4}$		−1	−1					-2^6				
18				5	$y_i^1-a_{i5}$			−1		−1			-2^9				
19				6	$y_i^1-a_{i6}$			−1					-1^{10}				
20				8	$y_i^1-a_{i8}$		−2		−1				-3^{11}				
21	2	9, 3	6*	ϕ	y_i^2	13	9	0	0	3	8	7					
22					$\Sigma_{j\in N_0^2}\bar{a}_{ij}$	−8	−35	−3		−13	−10	−6					
23	3	5	2		y_i^3	−2	4	−1	1	−3	0	0		1, 2, 4, 5, 6, 9	ϕ	7, 10	
24					$\Sigma_{j\in N_3}\bar{a}_{ij}$	−8		−3		−18							ϕ
25				3	$y_i^3-a_{i3}$							−2	-2^{13}				
26				8	$y_i^3-a_{i8}$	−2			−1		−2		-5^{14}				
27	4	5, 3	3	ϕ	y_i^4	6	14	0	0	1	0	−2		1, 2, 3, 4, 5, 6, 9	7, 10	8	
28					$\Sigma_{j\in N_3^4}\bar{a}_{ij}$			−2		−9							
29					$\Sigma_{j\in N_0^4}\bar{a}_{ij}$	−8	−10	−3	−1	−13	−3	−5					

Example 3. (Problem 4 of Tableau I).
The following is an ill-behaved problem (with no feasible solution):

$$4x_1+2x_2+x_3+5x_4+3x_5+6x_6+x_7+\ 2x_8+3x_9=\min,$$
$$3x_1+5x_2-2x_3-\ x_4\ \ -\ x_6\ \ -4x_8+2x_9\leqq -1,$$
$$6x_1-2x_2\ \ -2x_4+2x_5-4x_6+3x_7\ \ \leqq -3,$$
$$5x_2\ \ +3x_4-3x_5+6x_6-\ x_7\ \ -2x_9\leqq 2,$$
$$-5x_1-4x_2+\ x_3\ \ +5x_5\ \ -2x_7+\ x_8-\ x_9\leqq -8.$$

We do not reproduce in detail the computations but the application of the algorithm may be followed in Tableau IV, which shows the sequence of solutions and of the steps used at each iteration.

TABLEAU IV

Sequence of solutions		Sequence of steps	Sequence of solutions		Sequence of steps
s	J_s		s	J_s	
0	ϕ	A	16	3, 9	A
1	4	A	17	3, 9, 6	A
2	4, 9	A	18	3, 9, 6, 7	B
3	4, 9, 8	A	19	3, 9, 6, 7, 2	C (0)
4	4, 9, 8, 7	A	20	3, 9, 6, 1	C (2)
5	4, 9, 8, 7, 2	B	21	3, 7	B
6	4, 9, 8, 7, 2, 3, 5, 6	C (1)	22	3, 7, 2, 6	C (0)
7	4, 9, 8, 2	C (1)	23	3, 1, 2	C (0)
8	4, 9, 3	A	24	7	B
9	4, 9, 3, 7	A	25	7, 2, 6	C (0)
10	4, 9, 3, 7, 6	C (1)	26	8	A
11	4, 9, 3, 2	C (2)	27	8, 6	A
12	4, 7	A	28	8, 6, 9	A
13	4, 7, 6	C (1)	29	8, 6, 9, 1	C (3)
14	4, 1	C (1)	30	6	A
15	3	A	31	6, 9	D (2)
					Stop

The symbols in the Tableau indicate the following sequences of steps:

$$A = 1b,\ 2b,\ 3b,\ 8,$$

$$B = 1b,\ 2b,\ 3c,\ 4a,$$

$$C(k) = 1b,\ 2b,\ 3a,\ \underbrace{5b,\ 6a,\ \cdots,\ 5b,\ 6a,}_{2k}\ 5b,\ 6b,\ 8,$$

$$D(k) = 1b,\ 2b,\ 3a,\ \underbrace{5b,\ 6a,\ \cdots,\ 5b,\ 6a,}_{2k}\ 5a.$$

Example 4. (Problem 9 of Tableau I).

Another ill-behaved problem (with only one feasible solution) is the following:

$$\begin{aligned}
&5x_1+ x_2+ 3x_3+2x_4+ 6x_5+4x_6+7x_7+2x_8+4x_9+ x_{10}+ x_{11}+5x_{12}=\min,\\
&-x_1+3x_2-12x_3 - x_5+7x_6- x_7 +3x_{10}-5x_{11}- x_{12}\leqq -6,\\
&3x_1-7x_2 + x_4+ 6x_5 \leqq 1,\\
&11x_1 + x_3-7x_4 - x_6+2x_7+ x_8-5x_9 +9x_{11} \leqq -4,
\end{aligned}$$

$$\begin{aligned} & 5x_2+\ 6x_3 && -12x_5+7x_6 && +3x_8+\ x_9-8x_{10} && +5x_{12}\leqq 8, \\ -7x_1-\ & x_2-\ 5x_3+3x_4+\ && x_5-8x_6 && -2x_8+7x_9+\ x_{10} && -7x_{12}\leqq -7, \\ -2x_1 & && -4x_4 && -3x_7-5x_8-\ x_9 && +\ x_{11}+\ x_{12}\leqq -4. \end{aligned}$$

Tableau V shows the sequence of solutions and of the steps used at each iteration.

TABLEAU V

Sequence of solutions		Sequence of steps	Sequence of solutions		Sequence of steps
s	J_s		s	J_s	
0	ϕ	A	21	12, 4, 8	B
1	3	A	22	12, 4, 8, 2, 11	A
2	3, 4	A	23	12, 4, 8, 2, 11, 10	F (0)
3	3, 4, 8	A	24	12, 4, 11	C (1)
4	3, 4, 8, 10	A	25	12, 8	A
5*	3, 4, 8, 10, 12	E (0)	26	12, 8, 9	C (1)
6	3, 4, 8, 10, 2	F (0)	27	12, 7	F (1)
7	3, 4, 8, 10, 6	F (1)	28	8	A
8	3, 4, 12	C (0)	29	8, 4	A
9	3, 4, 6	A	30	8, 4, 7	C (0)
10	3, 4, 6, 10	A	31	8, 4, 6	H (0)
11	3, 4, 6, 10, 11	F (1)	32	8, 4, 1, 2	C (1)
12	3, 4, 2	A	33	8, 7	C (0)
13	3, 4, 2, 10	A	34	8, 2	C (0)
14	3, 4, 2, 10, 1	F (0)	35	8, 9	A
15	3, 4, 2, 5	F (1)	36	8, 9, 6	C (2)
16	3, 4, 1	F (0)	37	4	A
17	3, 8	G (0)	38	4, 6	C (1)
18	3, 7	C (1)	39	7	D (0)
19	12	A			Stop
20	12, 4	A			

The symbols A, B, $C(k)$ and $D(k)$ are used as in Tableau IV, while the other sequences of steps are:

$$E(k)=1a,\ \underbrace{5b,\ 6a,\ \cdots,\ 5b,\ 6a,}_{2k}\ 5b,\ 6b,\ 8,$$

$$F(k)=1b,\ 2a,\ \underbrace{5b,\ 6a,\ \cdots,\ 5b,\ 6a,}_{2k}\ 5b,\ 6b,\ 8,$$

$$G(k)=1b,\ 2b,\ 3c,\ 4b,\ \underbrace{5b,\ 6a,\ \cdots,\ 5b,\ 6a,}_{2k}\ 5b,\ 6b,\ 8,$$

$$H(k)=1b,\ 2b,\ 3a,\ \underbrace{5b,\ 6a,\ \cdots,\ 5b,\ 6a,}_{2k}\ 5b,\ 6c,\ 7a.$$

The optimal (in this case the only feasible) solution is the starred one, i.e.,

$$x_j = \begin{cases} 1, & (j = 3, 4, 8, 10, 12) \\ 0. & (j = 1, 2, 5, 6, 7, 9, 11) \end{cases}$$

ACKNOWLEDGMENTS

I AM indebted to PROF. WILLIAM W. COOPER, as well as to FRED GLOVER and to STANLEY ZIONTS for comments and suggestions which helped to improve this article. I also wish to acknowledge the help of ELENA MARINESCU, who carried out the computations for the forest-management problem discussed in the sixth section.

REFERENCES

1. G. B. DANTZIG, "Discrete Variable Extremum Problems," *Opns. Res.* **5**, 266–277 (1957).
2. H. M. MARKOWITZ AND A. S. MANNE, "On the Solution of Discrete Programming Problems," *Econometrica* **25**, 84–110 (1957).
3. K. EISEMANN, "The Trim Problem," *Management Sci.*, **3**, 279–284 (1957).
4. G. B. DANTZIG, "On the Significance of Solving Linear Programming Problems with Some Integer Variables," *Econometrica* **28**, 30–44 (1960).
5. A. CHARNES AND W. W. COOPER, *Management Models and Industrial Applications of Linear Programming*, Wiley, New York, 1961.
6. A. BEN-ISRAEL AND A. CHARNES, "On Some Problems of Diophantine Programming," *Cahiers du Centre d'Études de Recherche Opérationnelle* (Bruxelles) **4**, 215–280 (1962).
7. M. SIMONNARD, *Programmation linéaire*, Dunod, Paris, 1962.
8. G. B. DANTZIG, *Linear Programming and Extensions*, Princeton University Press, 1963.
9. E. BALAS, "Linear Programming with Zero-One Variables" (in Rumanian), *Proceedings of the Third Scientific Session on Statistics, Bucharest, December 5–7*, 1963.
10. ———, "Mathematical Programming in Forest Management" (in Rumanian), *Proceedings of the Third Scientific Session on Statistics, Bucharest, December 5–7*, 1963.
11. GH. MIHOC AND E. BALAS, "The Problem of Optimal Timetables," *Revue de Mathématiques Pures et Appliquées* **10** (1965).
12. R. E. GOMORY, "Outline of an Algorithm for Integer Solutions to Linear Programs," *Bull. Am. Math. Soc.* **64**, 3 (1958).
13. ———, "An All-Integer Programming Algorithm," in J. R. MUTH AND G. L. THOMPSON (eds.), *Industrial Scheduling*, Chap. 13, Prentice-Hall, 1963.
14. ———, "An Algorithm for Integer Solutions to Linear Programs," in R. L. GRAVES AND PH. WOLFE (eds.), *Recent Advances in Mathematical Programming*, pp. 269–302, McGraw-Hill, New York, 1963.

15. G. B. DANTZIG, "Note on Solving Linear Programs in Integers," *Naval Res. Log. Quart.* **6**, 75–76 (1959).
16. R. E. GOMORY AND A. J. HOFFMAN, "On the Convergence of an Integer-Programming Process," *Naval Res. Log. Quart.* **10**, 121–124 (1963).
17. E. M. L. BEALE, "A Method of Solving Linear Programming Problems When Some but Not All of the Variables Must Take Integral Values," Statistical Techniques Research Group, Technical Report No. 19, Princeton University, 1958.
18. A. H. LAND AND A. G. DOIG, "An Automatic Method of Solving Discrete Programming Problems," *Econometrica* **28**, 497–520 (1960).
19. J. F. BENDERS, A. R. CATCHPOLE, AND L. C. KUIKEN, "Discrete-Variable Optimization Problems," Paper presented to the Rand Symposium on Mathematical Programming, Santa Monica, 1959.
20. P. M. J. HARRIS, "The Solution of Mixed Integer Linear Programs," *Opnl. Res. Quart.* **15**, 117–133 (1964).
21. G. L. THOMPSON, "The Stopped Simplex Method, Part I," *Revue Française de Recherche Opérationnelle* **8**, 159–182 (1964).
22. ———, "The Stopped Simplex Method, Part. II," *Revue Française de Recherche Opérationnelle* 9 (1965).
23. P. L. IVANESCU, "Programmation polynomiale en nombres entiers," *Comptes Rendus de l'Académie des Sciences* (Paris) **257**, 424–427 (1963).
24. F. GLOVER, "A Bound Escalation Method for the Solution of Integer Programs," *Cahiers du Centre d'Études de Recherche Opérationnelle* (Bruxelles) **6**, (1964).
25. S. E. ELMAGHRABY, "An Algorithm for the Solution of the 'Zero-One' Problem of Integer Linear Programming," Department of Industrial Administration, Yale University, May 1963.
26. W. SZWARC, "The Mixed Integer Linear Programming Problem When the Variables are Zero or One," Carnegie Institute of Technology, Graduate School of Industrial Administration, May 1963.
27. F. LAMBERT, "Programmes linéaires mixtes," *Cahiers du Centre d'Études de Recherche Opérationnelle* (Bruxelles) **2**, 47–126 (1960).
28. S. VAJDA, *Mathematical Programming*, Addison-Wesley, 1961.
29. R. FORTET, "Applications de l'algébre de Boole en recherche opérationnelle," *Revue Française de Recherche Opérationelle* **4**, 17–25 (1960).
30. P. CAMION, "Une méthode de résolution par l'algèbre de Boole des problèmes combinatoires où interviennent des entiers," *Cahiers du Centre d'Études de Recherche Operationnelle* (Bruxelles) **2**, 234–289 (1960).
31. E. BALAS, "Un algorithme additif pour la résolution des programmes linéaires en variables bivalentes," *Comptes Rendus de l'Académie des Sciences* (Paris) **258**, 3817–3820 (1964).
32. ———, "Extension de l'algorithme additif à la programmation en nombres entiers et à la programmation nonlinéaire," *Comptes Rendus de l'Académie des Sciences* (Paris) **258**, 5136–5139 (1964).

33. F. Radó, "Linear Programming with Logical Conditions" (in Rumanian), *Comunicările Academiei RPR* **13**, 1039–1041 (1963).
34. 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).
35. P. Berthier and Ph. T. Nghiem, "Résolution de problèmes en variables bivalentes (Algorithme de Balas et procédure S.E.P.). Note de travail no. 33, Société d'Économie et de Mathématique Appliquées," Paris, 1965.

A NOTE ON THE ADDITIVE ALGORITHM OF BALAS†

Fred Glover

Carnegie Institute of Technology, Pittsburgh, Pa.

and

Stanley Zionts

Carnegie Institute of Technology and U. S. Steel Corp. Applied Research Laboratory, Monroeville, Pa.

(Received December 28, 1964)

IN THE preceding paper Egon Balas presents an interesting combinatorial approach to solving linear programs with zero-one variables. The method is essentially a tree-search algorithm that uses information generated in the search to exclude portions of the tree from consideration. The purpose of this note is: (1) to propose additional tests‡ meant to increase the power of Balas' algorithm by reducing the number of possible solutions examined in the course of computation; and (2) to propose an application for which the algorithm appears particularly well-suited. An acquaintance with Balas' paper is assumed in the discussion that follows.

First consider ways of reducing the number of solutions examined by Balas' method. In one approach to this objective, Balas defines the set D_s that (using his notation and equation numbers) consists of those $j \epsilon (N - C^s)$ such that, if $\boldsymbol{a}_j$ were introduced into the basis, the value of the objective function would equal or exceed the best value ($z^{*(s)}$) already obtained. It then follows immediately that the variables associated with elements of D_s may be ignored in seeking an improving solution along the current branch of the solution tree. However, D_s can be fruitfully enlarged by using a slightly less immediate criterion for inclusion, specified as follows. Each j in N_s is first examined—in any desired sequence—to see if

† The authors originally refereed Mr. Balas' paper for *Operations Research*. With their referee's report they submitted this manuscript extending some of his results.

‡ The additional tests proposed in this note reduce the number of solutions to be examined under the additive algorithm, at the expense of an increase in the amount of computation at each iteration. While we conjecture that on the balance it is worthwhile introducing these tests, this is a matter to be decided on the basis of computational experience.