Initialising Geoffrion's implicit enumeration algorithm for the zero-one linear programming problem

By J. L. Byrne[*] and L. G. Proll[†]

This paper describes a method of initialising Geoffrion's algorithm for the solution of linear programs in zero-one variables. The modification proposed involves three modules, one or all of which may be employed in a particular computation. A computational comparison is made between the original and modified algorithms.

(Received December 1968, revised April 1969)

1. Introduction

In a recent paper Geoffrion (1967) has proposed an implicit enumeration algorithm for the solution of the general linear programming problem in zero-one variables which is computationally attractive because of its simplicity and modest storage requirements compared with the related algorithms of Balas (1965) and Glover (1965). However, the computational experience with Geoffrion's algorithm provided by Freeman (1966) and Byrne (1967) shows that the time required to reach termination may be considerable even for relatively small problems. The aim of this paper is to show how this time may be considerably reduced by means of a modification of the non-iterative part of the algorithm. The modification proposed consists of three modules, one or all of which may be employed depending on the path along which the computation flows.

In the following sections the general linear programming problem in zero-one variables is referred to in the following form:

$$\begin{aligned} &\text{minimise } z = c'.x \\ &\text{subject to } Ax \geqslant b, \qquad\qquad (1)\\ &\qquad x_j = 0 \text{ or } 1, (j = 1, 2, \ldots, n), \\ &\qquad c \geqslant 0, \end{aligned}$$

where c, x are n-vectors, b is an m-vector and A is an m by n matrix. The coefficients in (1) are not restricted to being integers. Any linear programming problem in zero-one variables can be written in the form of (1) by means of a series of simple transformations (e.g. see Balas, 1965). The above form differs slightly from that considered by Geoffrion in the expression of the constraints but appears more natural to the authors.

2. Some aspects of Geoffrion's algorithm

In this section some aspects of Geoffrion's algorithm which are necessary to the development of the proposed modification are reviewed.

A solution of (1) is any binary n-vector. A solution x of (1) is feasible if $Ax \geqslant b$ and a feasible solution which minimises z over the set of feasible solutions is said to be optimal. The objective of an implicit enumeration algorithm is to obtain and verify an optimal feasible solution whilst explicitly enumerating as few as possible of the 2^n solutions of (1). This latter point is an important one since the experience of Freeman (1966) and Byrne (1967) shows that, in a number of cases, Geoffrion's algorithm may quickly find an optimal feasible solution but the subsequent verification that this solution is indeed optimal may take a considerable time.

The progress of Geoffrion's algorithm is governed by a sequence of ordered sets of indices, S, called partial solutions. For example, if $n = 5$ then $S = \{3, 5, -2\}$ implies that x_3, x_5, x_2 have been assigned values 1, 1, 0 in that order; the remaining variables x_1, x_4, are said to be free. A partial solution is determined from its successor by one of the two major steps of the algorithm, namely fathoming and augmentation. If an S can be fathomed, all solutions of (1) containing S have been implicitly enumerated and need no longer be considered; a new S is then constructed by backtracking which involves negating one positive element of S. If S cannot be fathomed, the value 1 is assigned to a variable, chosen, according to some criteria, from those free variables which decrease infeasibility in at least one constraint and which leave the cost of the current solution lower than the upper bound on z^*, the minimum value of z.

In the algorithm described in Geoffrion (1967), the initial partial solution is taken to be null and the augmenting variable is chosen so as to leave least total infeasibility, i.e. if the solution associated with S is defined by

$$\left.\begin{aligned} x_j^S &= 0, j \notin S \text{ or } -j \in S \\ x_j^S &= 1, j \in S \end{aligned}\right\} (j = 1, 2, \ldots, n)$$

$$\text{and} \qquad y_j^S = \sum_j a_{ij} x_j^S - b_i, (i = 1, 2, \ldots, m) \qquad (2)$$

then the chosen variable is that which yields

$$\max_j \left\{ \sum_i \min (0, y_i^S + a_{ij}) \right\}. \qquad (3)$$

Thus those variables, assigned the value 1, which are imbedded early in the sequence of partial solutions are those which make an apparently large contribution to feasibility without regard to cost. Considerable backtracking may therefore be necessary to change the assigned values of such variables with a consequent adverse effect on computation time. In addition, the feasibility measure (3) is calculated relative to the current partial solution so that, at each stage, only those variables which have previously been assigned a value are taken into account. However, assignments occurring later in

* *Department of Mathematics, University of Queensland*
† *Department of Mathematics, University of Southampton*

the sequence may render much of the contribution of the earlier assigned variables redundant in the sense that some constraints become oversatisfied. Thus variables which, earlier in the construction of S, appeared to be good contributors to feasibility may not now be so, given that the value 1 has since been assigned to other variables. Once a feasible solution has been constructed, this phenomenon can be detected and corrected for, e.g. consider the problem

$$\min 6x_1 + 4x_2 + 8x_3 + 3x_4 + 8x_5$$

$$\text{s.t.} \quad \begin{bmatrix} 23 & 35 & 28 & 11 & 32 \\ 30 & 20 & 13 & 7 & 15 \\ 17 & 5 & 22 & 9 & 19 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} \geqslant \begin{bmatrix} 60 \\ 30 \\ 50 \end{bmatrix}$$

Starting with $S = \phi$ and using (3) to select the augmenting variable, a feasible solution is obtained when $S = \{1, 5, 3\}$. Corresponding to the above partial solution,

$$y' = (23, 28, 8),$$

$$z = 22.$$

Comparison of the components of y with the coefficients of x_1 shows that, although x_1 appeared to be the best contributor to feasibility, the subsequent assignments, $x_5 = 1$, $x_3 = 1$, have rendered much of the contribution redundant. Inspection of the coefficients of the remaining free variables shows that x_1 could be replaced by x_4 to obtain

$$S = \{4, 5, 3\}$$

which also yields a feasible solution to the above problem and sets a lower upper bound on z^*.

The absence of cost in the augmentation criterion (3) has been noted by Geoffrion (1967) and Freeman (1966) but, as far as the authors are aware, no computational experience with alternative criteria has been published. The authors believe that the lack of an element of cost in the augmentation criterion is particularly disadvantageous in the initial stages of the algorithm since, in the absence of *a priori* information about the initial solution, an effective upper bound for z^* is set only when the first feasible solution is found. If (3) is used in the initial stages, the first feasible solution may well be an 'expensive' one so that until a better one is found, (i) it is relatively difficult to fathom S and (ii) the augmentation set is relatively large since the third of the required qualifications is trivially satisfied. The determination of a relatively low cost initial feasible solution may be expected, therefore, to be computationally advantageous.

In the tri-modular modification of Geoffrion's algorithm described here, two new measures of feasibility contribution are introduced. The first of these measures is used in the first module to construct an initial partial solution, if this is feasible then the second and third modules are applied. The second module attempts to sharpen the upper bound on z^* by eliminating redundant as ignments while, in the third module, the second measure of feasibility contribution is used to reorder the assignments.

3. Determination of an initial partial solution

Given a partial solution to (1), an assignment of 1 to any one of the free variables results in a contribution (which may be positive or negative) towards feasibility and in a new partial solution. The sequential feasibility factor (SFF), p_j, is proposed as a measure of this contribution corresponding to the current partial solution S for each free variable x_j and is defined by

$$p_j = \sum_{i \in I_-} \min (a_{ij}, -y_i^S) + \sum_{i \in I_+} \min (0, a_{ij} + y_i^S) \quad (4)$$

where

$$I_- = \{i : y_i^S < 0\}, I_+ = \{i : y_i^S \geqslant 0\}$$

and y_i^S is calculated from (2). The first sum in (4) is taken over the currently unsatisfied constraints and each term measures that part of the contribution of x_j which is necessary in order that the ith constraint is not oversatisfied, i.e. that part of the contribution of x_j which is redundant is discounted. The second sum in (4) is taken over the currently satisfied constraints for which the contribution of x_j to the ith constraint is redundant unless the assignment $x_j = 1$ makes the ith constraint unsatisfied when there is a negative contribution to

Table 1

Description of test problems

PROBLEM NUMBER	NUMBER OF CONSTRAINTS	NUMBER OF VARIABLES	% DENSITY OF NON-ZERO COEFFS. OF A	PROBLEM TYPE
1	10	15	96	Capital
2	10	20	94	Budgeting
3	10	28	94	Problems
4	10	28	94	
5	22	20	14	Storage
6	25	33	10	Problems
7	53	30	10	Determination of contents of extraction files

feasibility. Those variables for which the SFF is positive will increase overall feasibility if assigned the value 1 although possibly decreasing the feasibility of the solution for individual constraints, similarly variables with negative SFF tend to decrease overall feasibility and so it would seem reasonable not to consider such variables as candidates for a 1-assignment. Clearly, in the sense of contribution to feasibility, the free variables increase in importance as their SFF increases.

An initial partial solution may be constructed iteratively by reference to a sequence of entry coefficients e_j defined over the set of free variables. Definition of the entry coefficients involves a reference to the SFF and cost of each free variable. In the case $c_j > 0, (j = 1, 2, \ldots, n)$, the entry coefficients could be defined on the lines suggested by Glover (1965) and Petersen (1967) as the ratio of SFF to cost; the definition adopted here covers the case $c_j \geqslant 0, (j = 1, 2, \ldots, n)$. In constructing an initial partial solution, it is clearly hoped that a low cost feasible solution will be found. The entry coefficients should thus be chosen so as to rank those variables with zero cost and positive SFF before those with positive cost and positive SFF and should rank those variables with zero cost in order of their SFF's. By a suitable choice of scaling factor, (1) can be transformed so that

$$c_j = 0 \text{ or } c_j > 1, (j = 1, 2, \ldots, n). \tag{5}$$

Given (5), a set of entry coefficients having the desired properties is

$$e_j = \left\{ \begin{array}{ll} p_j - \sum_{i=1}^{m} y_i^S, & \text{if } c_j = 0 \text{ and } p_j \geqslant 0, \\ p_j/c_j & , \text{ if } c_j > 0 \text{ and } p_j \geqslant 0, \\ p_j & , \text{ if } p_j < 0. \end{array} \right\} \tag{6}$$

The first module consists of the following sequence of steps and yields an initial partial solution:

(*a*) set $S = \phi$,
(*b*) for all free variables x_j, calculate p_j, e_j from (4) and (6) respectively,
(*c*) determine a J such that

$$e_J = \max \{e_j\}.$$

If $e_J < 0$, exit from the initialisation routine and proceed with the usual steps of Geoffrion's algorithm. If $e_J \geqslant 0$, set

$$S = \{S, J\}$$

and recompute y^s.
(*d*) if $y^s \geqslant 0$, a feasible solution has been found and module 2 is entered with S as the partial solution. Otherwise, if any free variables remain, return to step (*b*); if not, exit from the initialisation routine and proceed with the appropriate steps of Geoffrion's algorithm with S as the partial solution.

4. Improvement of an initial feasible solution

If module 1 produces an initial partial solution yielding a feasible solution to (1), an attempt is made to derive a related feasible solution yielding a lower upper bound on z^*. This may be achieved by applying the concept of redundant contributions to feasibility, introduced above, in a simple manner.

Define $x_j^S = 1$ to be a redundant assignment if

$$y_i^S - a_{ij} \geqslant 0, (i = 1, 2, \ldots, m).$$

Similarly $x_j^S = 1$ is a replaceable assignment if there exists a free variable x_k such that

$$a_{ij} - a_{ik} \geqslant 0, (i = 1, 2, \ldots, m).$$

At each iteration of module 2, S is altered by that deletion, corresponding to a redundant assignment, or exchange, corresponding to a replaceable assignment, which gives the greatest reduction in cost. Module 2 terminates when no further reduction in cost is possible by such alterations.

5. Reordering of the initial partial solution

Apart from possible alterations incurred in module 2, S is still ordered in terms of the SFF of the appropriate variables. In module 3, S is reordered in terms of the second measure of feasibility contribution, namely the non-sequential feasibility factor, defined for each $j \in S$ by

$$q_j = \sum_{i=1}^{m} \max (0, a_{ij} - y_i^S). \tag{7}$$

Table 2

Computational results for original and modified algorithms

PROBLEM NUMBER	ORIGINAL ALGORITHM		MODIFIED ALGORITHM				z^*
	ITERATIONS	TIME	ITERATIONS	TIME	UPPER BOUND MODULE 1	UPPER BOUND MODULE 2	
1	179	6	112	4	1150	1150	1150
2	1851	77	585	27	2615	2535	2535
3	25963	1165	4730	245	3315	3315	3095
4	25545	1456	3274	217	7795	7795	7315
5	1293	75	874	55	323	323	323
6	19345	1800†	7971	770	215	215	191
7	45	14	27	14	104000	104000	104000

† Termination not reached, upper bound on $z^* = 224$ at premature termination.

The object of reordering S is to ensure that the first element in S to be changed at the backtracking stage corresponds to that variable which contributes least to feasibility according to (7). Thus, on exit from module 3,

$$S = \{k, l, \ldots, r\} \Rightarrow q_k \geqslant q_l \geqslant \ldots \geqslant q_r. \quad (8)$$

The appropriate feasible solution and the corresponding value of z are then recorded and the backtracking stage of Geoffrion's algorithm entered. Because of the structure of S, the amount of infeasibility associated with the next partial solution should be small and so the construction of a new feasible solution should be relatively easy. Additionally, those variables which contribute most to feasibility according to (7), will be the 'early' elements of S and hence may be expected to appear in many of the subsequent partial solutions. Again such partial solutions should be relatively easy to fathom.

6. Computational experience

The modifications described above were coded in ALGOL and incorporated in the authors' ALGOL code for Geoffrion's algorithm, the central part of which appears in Byrne and Proll (1968). The original and modified algorithms were then run on the ICL 1907 computer at Southampton University with several test problems, the main characteristics of which are listed in **Table 1**.

Test problems 1–4 are capital budgeting problems transformed into the form of (1); test problems 1, 2 and 3 correspond exactly to problems 3, 4 and 5 respectively in Petersen (1967), test problem 4 has the coefficient matrix and objective function of Petersen's problem 5 but has the right-hand side corresponding to Petersen's problem 4. Test problems 5 and 6 consist of constraints arising in the dynamic programming solution of a problem of purchasing and storing steel beams (Wiseman, 1968) together with random objective functions. Test problem 7 concerns a problem in data processing formulated on page 210 and uses the data given on page 212 of Seppala (1967). Test problems 5–7 have the sparse and highly structured coefficient matrices which frequently occur in practical problems.

Comparative results for the original and modified algorithms using the test problems are recorded in **Table 2** in terms of the number of iterations and the time taken, in seconds of cpu time, to termination. In addition the values of the upper bound for z^* attained in modules 1 and 2 of the modified algorithm are recorded against z^*.

Apart from problem 7, the results indicate that the modifications discussed in this paper result in a substantial improvement in the performance of Geoffrion's algorithm in both the number of iterations and the time to termination, time savings varying between 27% and 85%. Module 1 has produced a feasible solution in all cases and, at worst, one whose associated cost is within 13% of the minimum. Comparison of Table 2 and **Table 3** indicates that module 3 makes a significant contribution to the overall saving in several of the test problems.

Table 3

Computational results for the modified algorithm without module 3

PROBLEM NUMBER	ITERATIONS	TIME
1	112	4
2	733	34
3	5646	294
4	5004	342
5	874	55
6	7971	770
7	65	23

With the exception of problem 2, where its use resulted in the initial partial solution being optimal, module 2 has had no effect and this may be taken to indicate that its inclusion in the algorithm is not worthwhile. This may, however, be due to the quality of the solutions produced for the test problems by module 1. In principle, modules 2 and 3 can also be applied to any feasible solution and if an augmentation criterion which disregards cost is used, as in the current version of the algorithm, module 2 might be more useful. Work is in progress to establish ways of implementing this idea and to investigate alternative augmentation criteria.

Further experimentation is needed to establish limits on the size of practical problems which can be economically solved by the present code. However even for those problems which are too large to be run to termination, the modified algorithm can be expected to have recorded a good sub-optimal solution if the algorithm is prematurely terminated.

References

Balas, E. (1965). An Additive Algorithm for Solving Linear Programs with Zero-One Variables, *Opns. Res.*, Vol. 13, pp. 517–546.

Byrne, J. L. (1967). An Assessment of Recent Algorithms for the Zero-One Programming Problem, M.Sc. Dissertation, University of Southampton.

Byrne, J. L., and Proll, L. G. (1968). Algorithm 341: Solution of Linear Programs in Zero-One Variables, *CACM*, Vol. 11, p. 782.

Freeman, R. J. (1966). Computational Experience with a 'Balasian' Integer Programming Algorithm, *Opns. Res.*, Vol. 14, pp. 936–941.

Geoffrion, A. M. (1967). Integer Programming by Implicit Enumeration and Balas' Method, *SIAM Rev.*, Vol. 9, pp. 178–190.

Glover, F. (1965). A Multi-phase Dual Algorithm for the Zero-One Integer Programming Problem, *Opns. Res.*, Vol. 13, pp. 879–919.

Petersen, C. C. (1967). Computational Experience with Variants of the Balas Algorithm Applied to the Selection of R and D Projects, *Man. Sci.*, A 13, pp. 736–750.

Seppala, Y. (1967). Definition of Extraction Files and their Optimisation by Zero-One Programming, *BIT*, Vol. 7, pp. 206–215.

Wiseman, H. C. (1968). Two Applications of Mathematical Programming in the Trailer Industry, M.Sc. Dissertation, University of Southampton.