REVUE FRANÇAISE D'INFORMATIQUE ET DE RECHERCHE OPÉRATIONNELLE. SÉRIE VERTE

E. LOEHMAN
PH. T. NGHIEM
A. WHINSTON

Two algorithms for integer optimization

Revue française d'informatique et de recherche opérationnelle. Série verte, tome 4, n° V2 (1970), p. 43-63

<http://www.numdam.org/item?id=RO_1970__4_2_43_0>

R.I.R.O.
(4e année, V-2, 1970, p. 43-63)

TWO ALGORITHMS FOR INTEGER OPTIMIZATION (1)

by E. Lœhman, Ph. T. Nghiem and A. Whinston

Résumé. — *Cet article présente deux algorithmes d'exploration directe pour la résolution des programmes linéaires contenant des nombres entiers. Pour le premier algorithme, qui utilise la procédure d'énumération de Balas ; deux procédures de choix de la variable de branchement ont été expérimentées : dans la première, on choisit la variable à laquelle le programme linéaire continu donne la valeur la plus proche de* 0 *ou de* 1 *; dans la deuxième, on choisit la variable qui, forcée à la valeur* 0 *ou* 1, *conduit à l'accroissement minimum de la fonction économique après le premier pivot. Le deuxième algorithme utilise une méthode d'énumération qui généralise celle de Balas. Les temps de calcul sont donnés pour divers problèmes test.*

Various algorithms have been presented to solve integer programming problems. The most prominent methods are due to Gomory [7], Land and Doig [9], Balas [1], and Graves and Whinston [8]. Many other researchers have contributed to the development of this theory.

This paper presents two clearly related algorithms motivated by the ideas developed in Land and Doig. They developed a procedure which in its form was not particularly suitable for use on a computer. However, they introduced a key idea which has been used subsequently both implicity and explicitly by several algorithms. Their idea was to use the associated linear programming problem to guide the search for the optimal integer solution. The two algorithms to be presented below can be considered as computerized extensions of the original Land and Doig procedure.

Most linear integer programming problems of either theoretical interest or practical importance have the special property that the variables are restricted to the values zero and one. Furthermore, integer problems not possessing this property may be solved as zero-one problems using the well known binary representation. We confine ourselves to stating the algorithm when the variables must satisfy only the zero-one restrictions.

(1) Research supported in part by the Office of Naval Research and Army Research Office.

The problem to be solved can be stated as follows :

$$\text{Min} \sum_{j=1}^{N} c_j x_j + \sum_{j=N+1}^{m} c_j x_j$$

$$\text{s.t.} \sum_{j=1}^{m} a_{ij} x_j \leqslant b_i \qquad i = 1, 2, \ldots, m$$

$$x_j = 0,1 \qquad j = 1, 2, \ldots, N$$

$$x_j \geqslant 0 \qquad j = N + 1, \ldots, m$$

In effect the formulation allows for both integer constrained variables and continuous nonnegative variables. The coefficients of the problems are not required to be integer valued.

While both algorithms to be described in detail below use the associated linear programming problem they differ in several ways. Two different tree search procedures are used. The first method referred to as « inflexible bactracking » is essentially the search method presented by Glover [6], in his extension of the Balas Algorithm. The successive choices of integer variables and their initial assignment of a value of zero or one need not be predetermined but is directed by the search strategy. However, once a variable is selected and assigned an initial value the ordering of the variables is temporarily fixed. The second algorithm which is titled « a flexible tree search ennumerative procedure » does not make this requirement. Land and Doig presented still another tree search which however is not considered practical because of storage requirements for use on a computer.

The initial step of both algorithms is to find the optimal solution to the continuous linear programming problem where the zero-one integer variables are bounded between zero and one. The tree search successively assigns integer values to the variables. For each set of assigned integer values dictated by the tree search, the associated optimal linear programming solution in both algorithms is found by the dual simplex algorithm. Criterion of selection among the potential candidate variables differs somewhat between the two algorithms. The case of the dual simplex algorithm in integer programming has been suggested by Driebeek [4].

1. INFLEXIBLE BACKTRACKING TREE SEARCH ALGORITHM

Suppose there are N integer variables which can have the value zero or one. All possible combinations of zero and one are investigated by constructing a tree of partial solutions. Each node of the tree corresponds to a different partial solution. For this search the order in which variables are chosen and fixed to a value is important.

A node s is defined by :

1. A subset $J^s \subset \{ 1, 2, ..., N \}$

$$J^s = \{ j_1, \dots j_p \} \tag{1.1}$$

where the j_i are ordered.

2. To each j_i there is a subset $K^s_{j_i}$

$$K^s_{(j_i)} = \{ k^i_1, \dots k^i_l \} \tag{1.2}$$

and the k^i_j are ordered.

If $k \in K^s_j$, then $x_k = 0$ in the partial solution defined by node s.

If $k \in J^s$, then $x_k = 1$ in the partial solution defined by node s.

Let $K^s = \bigcup_{j_i} K^s_{j_i}$. The set $F = J^s \cup K^s$ defines the variables fixed at node s. The set $S = \{ 1, 2, ..., N \} - F$ contains the indices of the free variables. A new node may be obtained from node s by either a forward or a backward step.

Forward step from node s : A forward step consists of fixing a free variable to the values of zero or one. A new node s' is obtained as follows :

If S is not empty, f is chosen from S and the choice of $x_f = 0$ or 1 is made.

Replace by $S = S - \{ f \}$, so the set of free variables is reduced by one element.

1. If $x_f = 1$, then $J^{s'} = \{ j_i, ..., j_{p+1} = f \}$. The newly fixed variable is added to the last set of variables fixed to zero, and

$$K^{s'}_{j_i} = K^s_{j_i} \qquad i = 1, \dots p \quad K^s_{j_{p+1}} = \varnothing \tag{1.3}$$

2. If $x_f = 0$, then $K^{s'}_{j_p} = K^s_{j_p} \cup \{ f \}$ and

$$J^{s'} = J^s. \tag{1.4}$$

Backward step from node s : The backward step consists of setting a fixed variable at the alternate value. Let x_f be the variable from which backtracking occurs. The variables appearing below x_f in the tree are free.

1. If $f \in K^s_{j_p}$ (i.e., $x_f = 0$), then set $K^{s'}_{j_p} = K^{s'}_{j_p} - \{ f \}$, and

$$K^{s'}_{j_i} = K^s_{j_i} \quad \text{i} = 1, ..., p - 1 ; \; K^{s'}_{j_{p+1}} = \varnothing$$

$$J^{s'} = \{ j_i, ..., j_{p+1} = f \}. \tag{1.5}$$

The variable is removed from the set of variables fixed to zero and added to the set of variables fixed to one.

2. If $f = j_p$, (i.e., $x_f = 1$) then $J^{s'} = \{ j_i, ..., j_{p-1} \}$ and

$$K^{s'}_{j_{p-1}} = K^s_{j_{p-1}} \cup \{ f \}; \; K^{s'}_{j_i} = K^s_{j_i}, \; i = 1, p - 2. \tag{1.6}$$

The variable is removed from the set of variables fixed to one and added the set of variables fixed to zero.

By forward steps and backward steps, all nodes (and hence, all partial solutions) can be enumerated.

EXAMPLE :

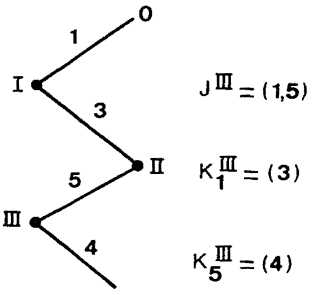


Figure 1

Figure 1 shows the tree structure for a partial solution at node III.

Forward Step : If we go forward from x_4 and choose, say, $x_6 = 1$, then we obtain a new node IV.

$$J^{IV} = \{ 1, 5, 6 \} \quad K_6^{IV} = \varnothing$$

If we choose $x_6 = 0$, then

$$J^{IV} = \{ 1, 5 \}$$

$$K_1^{IV} = \{ 3 \}, K_5^{IV} = \{ 4, 6 \}$$

In either case, we remove $\{ 6 \}$ from the set of free variables.

Backward Step : If we go backward from x_4, then we set x_4 at its complementary value, i.e., $x_4 = 1$.

$$J^{V} = \{ 1, 5, 4 \}$$

$$K_1^{V} = \{ 3 \}, K_5^{V} = \varnothing, K_4^{V} = \varnothing$$

and all variables appearing below x_4 are freed.

This tree search method is used in the first zero-one algorithm. The problem considered is to :

(1.7) minimize $z_0 = c'x$ subject to $Ax \leqslant b$, $x \geqslant 0$ and zero-one solutions for $x_1, \ldots x_N$ are required

while $x_{N+1}, \ldots x_n$ may have any non negative value. The initial tableau is shown below.

Value Basic Variables	Basic Variables	Integer Variables	Continuous Variables
V.B.V.	V.B.	$x_1 \dots x_N$	$x_{N+1} \dots x_n$
b_1	y_1	$a_{11} \dots a_{1N}$	$a_{1N+1} \dots a_{1n}$
.	.	. .	. .
.	.	. .	. .
.	.	. .	. .
b_m	y_m	$a_{m1} \dots a_{mN}$	$a_{mN+1} \dots a_{mn}$
0	z_0	$-c_1 \dots -c_N$	$-c_{N+1} \dots -c_n$

Figure 2

If initially some $(-c_j) > 0$, where x_j is an integer variable, then we make a transformation of the original tableau as follows :

$$
\begin{aligned}
b_i &\text{ is replaced by } b_i - a_{ij} && i = 1, \dots m \\
a_{ij} &\text{ is replaced by } -a_{ij} && i = 1, \dots m \\
z_0 &\text{ is replaced by } z_0 + c_j \\
-c_j &\text{ is replaced by } +c_j
\end{aligned}
\tag{1.8}
$$

After this transformation, the j^{th} column now corresponds to the variable $1 - x_j$.

If some $(-c_j) > 0$, where x_j is a continuous variable, then we must add one additional constraint :

$$x_1 + x_2 \dots + x_n \leqslant b_{m+1}, \tag{1.9}$$

where b_{m+1} is large enough so that the addition of this constraint will not change the solution to the original system. By pivoting once, we obtain a feasible solution to the dual problem. The pivot will occur in row $(m + 1)$ and the column is determined by $\max_j (-c_j)$.

After performing these steps, we have $z_j \leqslant 0$ $j = 1, \dots n$ where, z_j denotes the elements in the last row of the tableau, so the solution to the linear program (L.P.) can be obtained by using thed ual method, i.e., we begin with a feasible, but not optimal, solution to the dual problem and an optimal but not feasible solution to the primal problem.

The dual method with bounds is described by Wagner [11]. Briefly the method is to find the negative b_i which is least, say b_l. The variable in

the basis in row l will be removed from the basis. Then, $\min_{a_{lj}<0} z_j/a_{lj}$ is found, say $\frac{z_k}{a_{lk}}$. The variable corresponding to column k will be introduced into the basis in row l.

After pivoting, we check that all b_i corresponding to integer variables are less than or equal to one. Suppose $b_l > 1$ and x_{i_l} corresponding to b_l is an integer variable. Then, we introduce $\bar{x}_{i_l} = (1 - x_{i_l})$ into the tableau. The result is, we replace :

$$b_l \text{ by } 1 - b_l$$

(1.10) and

$$a_{lj} \text{ by } -a_{lj} \qquad j = 1, \ldots n$$

Note that $1 - b_l < 0$ so that the primal problem is infeasible. However, because of our pivoting rules, $z_j \leqslant 0$, $j = 1, \ldots n$. Thus, we pivot according to the dual rules again.

The process terminates if there are no negative a_{ij} in a row with $b_i < 0$ (infeasibility) or if all b_i are nonnegative (optimal, feasible solution to the primal).

After the tableau with $z_j \leqslant 0$, $j = 1, \ldots n$ has been set up, then we are ready to begin the zero-one algorithm.

	Value Basic Variables	Basic Variables	Integer Variables	Continuous and slack Variables
			$x_1 \ldots x_l$	$x_{l+1} \ldots x_n$
Integer Variables	b_1	y_1	$a_{11} \ldots a_{1l}$	$a_{1l+1} \ldots a_{1n}$
	$\cdot$	$\cdot$	$\cdot \quad \cdot$	$\cdot \quad \cdot$
	$\cdot$	$\cdot$	$\cdot \quad \cdot$	$\cdot \quad \cdot$
	$\cdot$	$\cdot$	$\cdot \quad \cdot$	$\cdot \quad \cdot$
	b_j	y_j	$a_{j1} \ldots a_{jl}$	$a_{jl+1} \ldots a_{jn}$
	b_{j+1}	y_{j+1}	$a_{j+1,1} \quad a_{j+1,l}$	$a_{j+1,l+1} \quad a_{j+1,n}$
	$\cdot$	$\cdot$	$\cdot \quad \cdot$	$\cdot \quad \cdot$
	$\cdot$	$\cdot$	$\cdot \quad \cdot$	$\cdot \quad \cdot$
	$\cdot$	$\cdot$	$\cdot \quad \cdot$	$\cdot \quad \cdot$
	b_m	y_m	$a_{m1} \ldots a_{ml}$	$a_{m,l+1} \ldots a_{mn}$
	f	z_0	$z_1 \ldots z_l$	$z_{l+1} \ldots z_n$

Figure 3

In *step zero* of the algorithm, we find an optimal solution to the linear programming (L.P.) problem. This gives us a tableau in canonical form as shown in figure 3

$$\begin{aligned} &\text{with } b_i \geqslant 0 && i = 1, \ldots m \\ &\quad 0 \leqslant b_i \leqslant 1 && i = 1, \ldots j\,;\, j + l = N \\ &\quad z_i \leqslant 0 && i = 1, \ldots n. \end{aligned}$$

This tableau is obtained by reordering the variables and constraints. The variables $x_1, \ldots x_l$ are non-basic integer variables and $x_{l+1}, \ldots x_n$ are non-basic continuous or slack variables. Similarly, $y_1, \ldots y_j$ are required ot become integer. Here, corresponding to integer variables, we may have either x_i or $\bar{x}_i$, where $\bar{x}_i = (1 - x_i)$, and y_i or $\bar{y}_i$ where $\bar{y}_i = (1 - y_i)$.

After solving the L.P., $x_1, \ldots x_l$ have integer values. It has been observed that many of these will be at their optimum value, while only a few will have the wrong value. For instance, in a problem due to Bouvier and Messoumian [3] which has twenty integer variables and twenty constraints, thirteen variables have zero-one values in the L.P. solution and of these only two are different from their optimum value. This suggests that we begin the forward step at level $l + 1$ of the tree. $x_1, \ldots x_l$ will be fixed but not ordered so at this point, we are really considering a set of l factorial trees.

Later, in backtracking, an order will have to be chosen for $x_1, \ldots x_l$. This is done by finding which variable, when changed to the complemen-

	Value Basic Variables	Basic Variables	Fixed integer	Free integer	Continuous and slack
			$x_1 \ldots x_k$	$x_{k+1} \ldots x_L$	$x_{L+1} \ldots x_n$
Integer variables	b_1	y_1	$a_{11} \ldots\ldots$	$\ldots\ldots\ldots$	$\ldots\ldots a_{1m}$
	$\vdots$	$\vdots$	$\vdots$		$\vdots$
	b_j	y_j			
	b_n	y_n	$a_{m1} \ldots\ldots$	$\ldots\ldots\ldots$	$\ldots\ldots a_{mn}$
	f	z_0	$z_1 \ldots\ldots$	$\ldots\ldots\ldots$	$\ldots\ldots z_n$

Figure 4

The tableau at the kth level

tary value, will result in the least increase in the objective function. For example, if x_j appears as a non-basic integer variable (and, hence, is zero) and if we transform the tableau to have $x_j = 1$, then we substitute $(1 - \bar{x}_j)$ in the appropriate column and the criterion function is increased from z_0 to $z_0 - z_j$. Thus, to find the variable corresponding to least increase in z_0, we find $\min_j (-z_j)$.

Step 1. We begin the forward step starting from level $l + 1$ of the tree.

k th *Stage.* Figure 4 is obtained by reordering the variables and constraints and shows the situation after either a forward or backward step a level k of the tree of partial solutions.

a) *Forward step.* The forward step occurs in going down the tree. At the beginning of the kth stage, $(k - 1)$ variable have fixed values of 0 or 1, and we are not allowed to pivot in the columns corresponding to these variables. From the list of basic integer variables we choose a variable to force out of the basis at a fixed value of zero or one. Either a free, a continuous, or a slack variable may then be introduced in the basis in its place.

b) *Backward step.* Let x_k be the variable fixed at level k. Backtracking occurs either with infeasibility for either $x_k = 0, 1$ or if not both values 0, 1 have been tried. Variables in the tree occurring below x_k are free. x_k is set to the complement of its previous value. If this is infeasible, we go to level $k - 1$ and the backward step occurs again. Otherwise, the forward step is performed.

Stage N : When no integer variables appear in the basis, then all are fixed to zero or one. Begin backtracking. Continue the backward and forward steps until all solutions have been enumerated.

At each forward step, it is necessary to choose the next variable to introduce and to decide if it should be introduced as zero or one. Two choice procedures are investigated.

One is essentially just a rounding off procedure. The list of integer variables is considered. Those free integer variables which are not in the basis (hence, already have integer values) are first fixed. Then, the variable which is closest to being integer is selected to leave the basis and zero or one is chosen as its value, depending on which number is closer.

Another procedure is based on considering the tableau and choosing the variables which leave and enter the basis to obtain the smallest increase in the criterion function. Integer variables must be forced out of the basis in order to be fixed. If there are some variables in the basis which are free integer variables, we consider :

$$RP = \min_j (-z_j\ a_{ij})\ b_i \qquad \text{for } a_{ij} > 0 \tag{1.11}$$

and

$$RN = \min_j (-z_j\ a_{ij})\ (b_i - 1) \qquad \text{for } a_{ij} < 0 \tag{1.12}$$

for each row i corresponding to an integer variable in the basis.

Suppose we were to pivot in row i to fix the variable in the basis in that row to zero or one. The following rules for fixing that variable to zero or one results in the least increase in the criterion function. We would find $R = \min(RP, RN)$ for row i.

If $R = RP$, then if the variable in the basis in row i is an original variable, we force it to zero.

If $R = RN$ and the variable in the basis in row i is an original variable, we force it to one.

Likewise, if $R = RP$ and the variable is a complementary one, we force the original variable to one.

If $R = RN$ and the variable appearing in row i is complementary, we force the original variable to zero.

These rules come about because of the tableau transformation involved when a variable is forced out of the basis and fixed at zero or one. Then, if we were to pivot in row i, the objective function would be increased from z_0 to $z_0 + \min(RP, RN)$. We would obtain the smallest increase which would occur if the variable in row i were fixed to zero or one.

The ratios RP and RN are considered for each row corresponding to an integer variable. We find the row resulting in the smallest possible increase in z_0 and remove the variable in that row from the basis, fix it to zero or one, and introduce the variable which corresponds to $\min(RP, RN)$ in that row.

Forward steps, backward steps, and freeing fixed variables all involve transformations of the tableau in figure 4. The rules for these transformations are as follows :

1. Forward step

As outlined above, we choose a variable to remove from the basis.

(i) If the variable chosen is an original variable which appears in the basis in row l :

a. to set it to zero multiply the corresponding row by (-1) i.e., replace a_{lj} by $-a_{lj}$, $j = 1, \ldots n$ and b_l by $-b_l$;

b. to set it to one substitute $(1 - \bar{x}_{it})$ for the variable where it appears. This results in replacing b_l by $b_l - 1$.

(ii) If the variable chosen is complementary to the original variable and appears in the basis in row l :

a. to set the original variable to zero substitute $(1 - x_{it})$ for $\bar{x}_{it}$. Then b_l is replaced by $b_l - 1$;

b. to set the original variable to one, multiply the corresponding row by (-1).

In the abose cases, the number in row l in the « value basic variables » column is now negative so if it is feasible, the dual pivoting rules result in the removal of the basic variable in row l from the basis (1).

2. Backward step : A variable is fixed at zero or one, and we want to try the complementary value. Recall fixed variables are non-basic.

(i) $\bar{x}_{ik}$ appears in the tableau in column k, so $x_{ik} = 1$. To set $x_{ik} = 0$ in column k, subsititute $(1 - x_{ik})$ for x_{ik}. This results in :

$$
\begin{aligned}
&b_i \text{ is replaced by } b_i - a_{ik},\\
&a_{ik} \text{ is replaced by } -a_{ik} \qquad i = 1, \ldots, m,\\
&z_0 \text{ is replaced by } z_0 - z_k\\
&z_k \text{ is replaced by } -z_k.
\end{aligned}
\tag{1.13}
$$

Then, if the resulting tableau is feasible, or a feasible tableau is obtained after pivoting, we will have $x_{ik} = 0$.

(ii) x_{ik} appears in the tableau in column k, so $x_{ik} = 0$. To set $x_{ik} = 1$ in column k substitute $(1 - \bar{x}_i)_k$ for x_{ik}. The transformations are as above, and if feasibility is obtained, we will have $x_{ik} = 1$.

3. Freeing a fixed variable

If a variable is fixed, then it is not allowed to re-enter the basis. This means that no element a_{ij} in the column corresponding to the variable may be used as a pivotal element. Therefore, when pivotal elements are chosen, only those ratios $\dfrac{z_j}{a_{ij}}$ corresponding to free variables are considered. For this reason, some z_j corresponding to fixed variables may become positive through pivoting. When a variable is freed after being fixed, a_{ij} in the column corresponding to the variable may again be used as pivotal elements. In order that the dual pivoting rules not be violated, the sign of the z_j corresponding to the variable to be freed is checked. If

---

(1) Note : Suppose the equation in row i is

$$a_{i1}x_1 + \ldots + a_{in}x_n + \bar{y}_i = b_i.$$

Then, if we multiply this row by (-1) we have

$$-a_{i1}x_1 \ldots - a_{in}x_n \ldots - y_i = -b_i.$$

So in pivoting, we must remember that the coefficient of the basic variable is now -1.

Likewise, if we substitute $(1 - y_i)$ for the basic variable y_i we have :

$$a_{i1}x_1 + \ldots + a_{in}x_n + (1 - \bar{y}_i) = b_i$$

or

$$a_{i1}x_1 + \ldots + a_{in}x_n - \bar{y}_i = b_i - 1.$$

it is positive, then the following transformation must be made before the variable can be freed :

(i) If x_{ik} appears in the tableau in column k substitute $(1 - \bar{x}_i)_k$ for x_{ik} column k. We replace :

$$\begin{aligned} & b_i \ by \ b_i - a_{ik} \\ & a_{ik} \ by \ -a_{ik} \\ & z_0 \ by \ z_0 - z_j \\ & z_j \ by \ -z_j \end{aligned} \tag{1.14}$$

(ii) If $\bar{x}_{ik}$ appears, substitute $(1 - x_i)_k$ for $\bar{x}_{ik}$ and perform the transformation as above.

In either case, z_j is now negative as it should be for the dual procedure.

Forward steps may be performed as long as the tableau in figure 4 is feasible. When infeasibility occurs, transformations must be performed on the tableau in figure 4 until a feasible tableau is again obtained. Since the dual method is being used, pivoting is only allowed where $a_{ij} > 0$. As certain variables become fixed, we are no longer allowed to pivot in the columns corresponding to these variables so that the choices of pivots are reduced as we proceed down the tree.

For the dual method, we begin with optimality for the free variables since $z_j \leqslant 0$ corresponding to the free variables but not with feasibility since some b_i may be negative. Feasibility is achieved when all $b_i \geqslant 0$ and the pivoting rules assure that all z_j corrssponding to free variables remain non-positive. If at some point there are negative b_i but no negative a_{ij} on which pivoting is allowed, then we have infeasibility. In the program, if an infeasibility occurs, we cease pivoting to fix the current chosen variable to 0 or 1. The last infeasible tableau is used and transformed in the following steps. Backtracking occurs until a feasible tableau is obtained.

The tree search method given above shows how all partial solutions may be enumerated. However, it is time consuming and unnecessary to consider all partial solutions. Some may be eliminated by various truncation procedures. The truncation procedure used here is based on a principle of Land and Doig [9]. In the tree, we begin from z_0, the optimum feasible solution to the L.P. As we fix variables at integer values, the solution set becomes smaller and the value of the criterion function becomes larger since the problem is one of minimization. The solution to the L.P. is a lower bound for the integer solutions. Figure 5 below shows several branches of a typical solution tree.

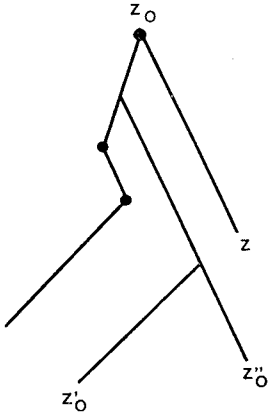


Figure 5

When one path through the tree is completed, we obtain a value of the criterion function z_0' In constructing each subsequent path at each tableau transformation and pivot, we consider the value of the criterion function z. If at some point, z is greater than z_0' then further pivots will only increase the criterion function z. Hence, the value of the criterion function which would be obtained at the end of this path will be bigger than z_0'. It is, thus fruitless to consider this branch further, so we backtrack and consider a different path. If we reach the bottom of the tree and $z_0'' < z_0'$. then we have found a better solution and z_0'' is used in future comparisons.

In writing a computer program for this algorithm, the main problems are ones of bookkeeping. That is, we must keep track of which variables are free, which have been fixed and what value they have. Also, it is necessary to know whether the variable or its complement appears in the tableau. Finally, we have to know which columns correspond to which non-basic variables and which basic variables appear in which rows. All this information is stored in various arrays. Three arrays are used to enumerate the partial solutions. At level K, $IL(l)$, ... $IL(K)$ give the indices of the fixed integer variables in order. With each level I, we associate an array $R(I)$.

$R(I) = 0$ means that both zero and one have been tried for the variable assigned to level I.

$R(I) = 1$ means that one has already been tried for the variable at level I.

$R(I) = 2$ means zero has been tried.

$R(I) = 3$ means that both zero and one should be tried for the variable at level I, that is the variable at level I is free.

If $R(I) = 0$, $I = 1, \ldots N$, then the algorithm terminates since all possible solutions have been checked.

The array S gives the indices of the free variables. In a forward step, if variable I is chosen, I is removed from S. In a backward step, the index of the variable freed is added to S. Thus, the numbers in S give the possible choices of variables at each level and at level K, there are $N - K$ elements in S.

In the program, an array BD is used to indicate whether a variable or its complement appears.

$BD(J) = \quad 1$ if x_j appears,

$BD(J) = -1$ if $\bar{x}_j$ appears.

The following arrays are also used :

$IN(I)$ tells which variable appears in the basis in row I.
$KS(J)$ tells which variable appears in the basis in column J.
$IROW(J)$ tells which row variable J is in if it is in the basis.
$KOL(K)$ tells which column variable K is in if it is non-basic.

The following flow chart summarizes this algorithm in terms of the computer program.

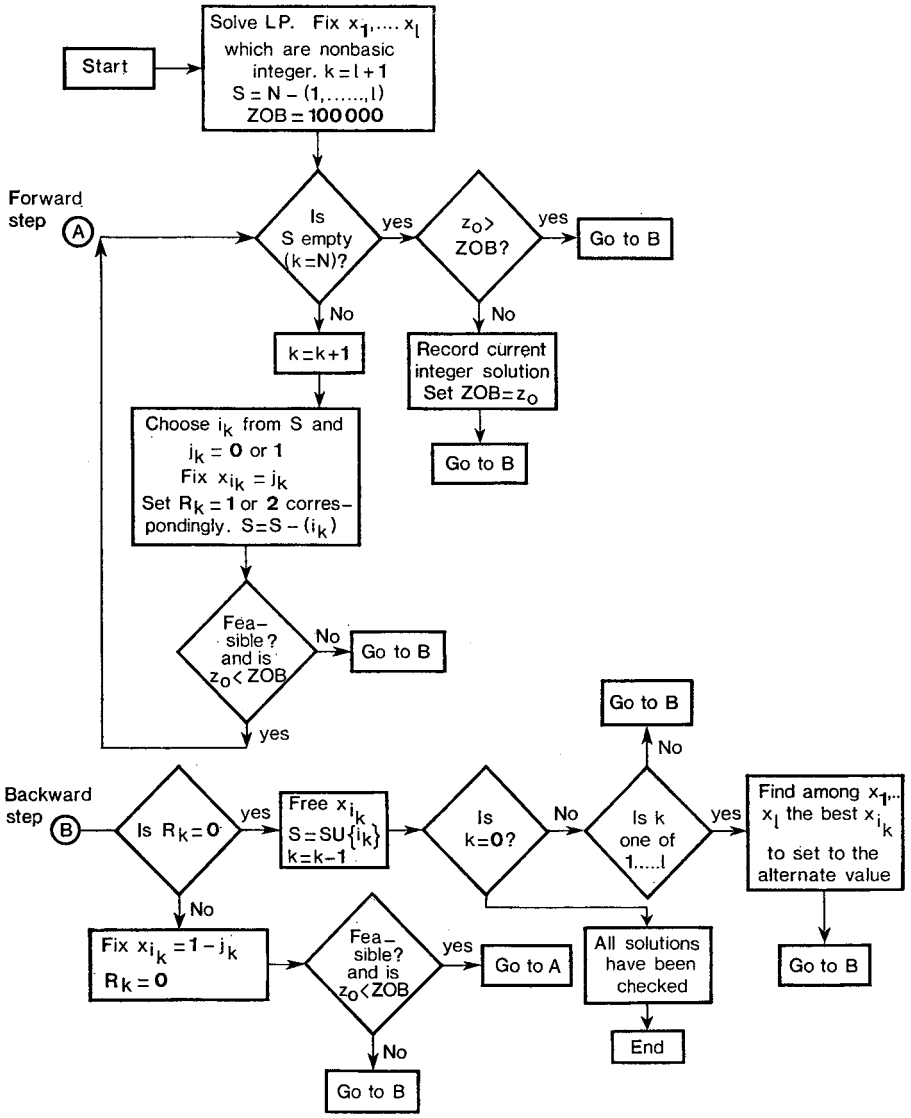


Figure 6

2. A FLEXIBLE TREE SEARCH PROCEDURE [10]

As before, we consider all possible combinations of zero and one for N variables. This algorithm differs from the former in that the order in which variables appear is not fixed in backtracking and variables are first fixed to one, and then to zero. These factors result in a different definition of a node from the one presented in Section 1. A node is characterized by strings of variables set one and each string of ones is followed by a set which contains at most one variable set to zero.

Formally, the characteristics of a node s are given by :

1. $J^s = \{ j_1, j_2, \ldots j_p \}$, where J^s a subset of the integers $(1, \ldots, N)$ and J^s is partitioned into

$$J^s = J_1^s \cup J_2^s \ldots \cup J_h^s \tag{2.1}$$

where the subscript i denotes the rank or index of the subset J_i^s. h, the highest rank of the subsets of J_i^s is called the rank of node s. The sets J_i^s are strings of variables which are fixed to one at node s. The elements within each J_i^s are not ordered. Some of the J_i^s may be empty.

2. With each subset J_i^s we associate a subset K_i^s which either contains one elements k_j^s or is empty. If K_i^s is not empty, then it contains the index of a variable which is fixed to zero. The ordering of J_i^s induces an ordering of the K_i^s. Let

$$K^s = \bigcup_{i=1}^{h} K_i^s \qquad \text{and} \tag{2.2}$$

$$F = K^s \cup J^s \qquad (\text{Note that } K^s \cap J^s = \varnothing) \tag{2.3}$$

Then, $S = \{ 1, 2, \ldots, N \} - F$ is the set of « free variables » which have not yet been fixed to zero or one. If K_h^s associated with J_h^s is empty, then node s is said to be open and in forward steps, elements may be added to the set J_h^s.

If K_h^s is not empty, then node s is said to be *closed* and the indices of variables fixed to one in succeeding forward steps will be placed in a set $J_{h+1}^{s'}$. s' denotes a new node defined by

$$\begin{aligned} J_i^{s'} &= J_i^s \qquad i = 1, h \\ K_i^{s'} &= K_i^s \qquad i = 1, h \\ J^{s'} &= J_1^{s'} \cup \ldots \cup J^{s'} \cup J_{h+1}^{s'} \end{aligned} \tag{2.4}$$

Consider the partial solution given in figure 7.

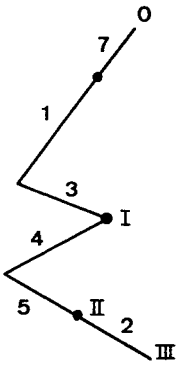


Figure 7

The numbers indicate the indices of variables fixed to 0 or to 1 at a node. Suppose we are at node III of the tree. Then,

$$J_1^{\text{III}} = \{ 7, 1 \}$$
$$K_1^{\text{III}} = \{ 3 \}$$
$$J_2^{\text{III}} = \{ 4 \}$$
$$K_2^{\text{III}} = \{ 5 \}$$
$$J_3^{\text{III}} = \varnothing$$
$$K_3^{\text{III}} = \{ 2 \}$$

Since K_3^{III} is not empty, node III is closed. If next we were to choose variable 6 to fix to one, then would go in a new node IV.

$$J_i^{\text{IV}} = J_i^{\text{III}} \qquad i = 1, 2, 3$$
$$K_i^{\text{IV}} = K_i^{\text{III}} \qquad i = 1, 2, 3$$
$$J_4^{\text{IV}} = \{ 6 \}$$

and node IV is open since $K_4^{\text{IV}} = \{ \varnothing \}$.

As before, the partial solutions are enumerated by forward and backward steps. Let h be the rank of node S. A *forward step* is possible if S, the set of free variables, is not empty. Then we choose an f from S and variable x_f is set to one.

1. If node s is open, then the rank of the resulting node is h.

$$J_h^s = J_h^s \cup \{ f \} \tag{2.5}$$

2. If node s is closed, then the rank of the resulting node s' is $h + 1$, and

$$
\begin{aligned}
J^{s'}_{h+1} &= \{ f \} \\
K^{s'}_{h+1} &= \varnothing \\
J^{s'}_{i} &= J^{s}_{i} \qquad i = 1, \ldots h \\
K^{s'}_{i} &= K^{s}_{i} \qquad i = 1, \ldots h.
\end{aligned} \tag{2.6}
$$

A *backward step* is possible only if J^s is not empty.

Let J^s_g be the subset of highest order which is not empty. By some decision rule we choose a k from J^s_g and switch x_k to zero. Then k is removed from J^s_g. A new node s' is then obtained where

$$
\begin{aligned}
J^{s'}_{i} &= J^{s}_{i} \qquad i = 1, \ldots g - 1 \\
J^{s'}_{g} &= J^{s}_{g} - \{ k \} \\
K^{s'}_{i} &= K^{s}_{i} \qquad i = 1, \ldots g - 1 \\
K^{s'}_{g} &= \{ k \}
\end{aligned} \tag{2.7}
$$

and

$$
\begin{aligned}
K^{s'}_{g+1} &= \varnothing \\
&\vdots \\
K^{s'}_{h} &= \varnothing
\end{aligned}
$$

i.e., the elements in $K^{s'}_{g+1}, \ldots K^{s'}_{h}$ are freed.

Consider our previous example again at node III. Since J^{III}_2 is the first non-empty subset ($J^{\mathrm{III}}_3 = \varnothing$) $g = 2$. Since $J^{\mathrm{III}}_2 = \{ 4 \}$, we choose x_4 to backtrack on. We then set $x_4 = 0$.

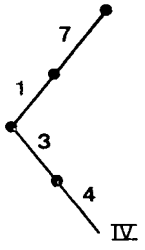


Figure 8

The resulting partial solution corresponding to the new node IV obtained is given in figure 8: and we have

$$
\begin{aligned}
J^{\mathrm{IV}}_1 &= \{ 7,1 \} \\
J^{\mathrm{IV}}_2 &= \varnothing \\
K^{\mathrm{IV}}_1 &= \{ 3 \},\ K^{\mathrm{IV}}_2 = \{ 4 \}
\end{aligned}
$$

Now since we have changed the value of x_4, perhaps it is possible for x_5 and x_2 to have alternate values. Thus, x_5 and x_2 are freed and allowed to become either 0 or 1.

With this tree search procedure, we again consider the problem given in (1.7). We first solve the linear programming problem with the

integer variables only constrained to be between zero and one. As before the tableau at this point is given in figure 3. The tree search algorithm may be summarized as follows :

0. Start. Solve L.P.

All variables are « free ». Rank $h = 1$.

1. Proceed to following partial solution. *Forward move :* Choose f from S and x_f will be set to 1. Add x_f to J_h.

2. Is this choice feasible ? If so, go to 4. If not, then go to 3.

3. *Backward move :*

Choose a k from J_g, where J_g is of highest rank of the non-empty subsets. Then $K_g = \{ k \}$ and the current node is closed. If there are variables of rank higher than g, they are freed. In subsequent steps, the rank $h = g + 1$. x_k is set to 0 and fixed. Go to 2.

4. Check for next move.

If an integer solution has been achieved, go to 5.

If not go to 1.

5. If all possible solutions have been tried, go to 6.

Otherwise go to 3.

6. Stop.

The choice procedures for this algorithm are simpler than the ones for the previous algorithm since we always first try to set a variable to 1. If this is not feasible, then the value 0 is tried. There are choice procedures for both forward moves and backward moves since in backtracking we may choose which of the previously fixed variables are to be set to the alternate value.

Forward move : We find which basic integer variable when removed from the basis and fixed at one results in the least increase in the criterion function. The choice rules are the same as those given in Section 1 except here we are only considering fixing variables to one.

Backward move : The highest order non-empty J_h is found. From this set of variables fixed to one, we choose a variable to fix to zero by considering z_j for all $j \in J_h$. We find the maximum of these, say z_k. The column of this variable is then transformed by replacing

$$\begin{array}{llll} b_i & \text{by} & b_i - a_{ik} & i = 1, \ldots m \\ a_{ik} & \text{by} & -a_{ik} & \\ z_0 & \text{by} & z_0 - z_k & \\ z_k & \text{by} & -z_k & \end{array}$$

Since z_k is the largest, we get the smallest increase (if z_k is negative) or the biggest decrease (if z_k is positive) in z_0 by fixing the corresponding variable to zero.

Two other tableau transformations are freeing variables fixed to zero and fixing variables to one. These rules are the same as those given in Section 1.

Figure 9 summarizes the algorithm and shows how the computer program is organized.

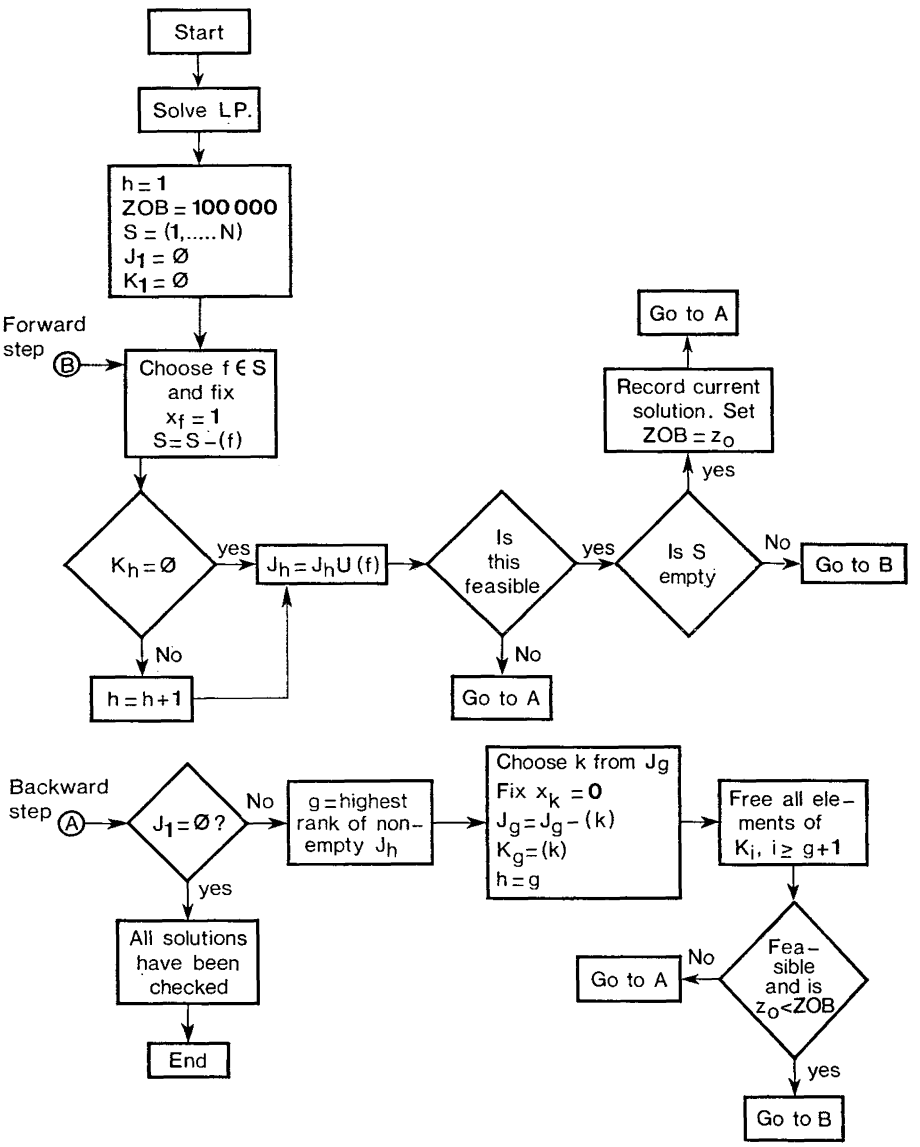


Figure 9

3. COMPUTATIONAL RESULTS

Some zero-one problems due to Bouvier and Messoumian [3] were run on the CDC 6500 using the algorithms above. The results are given below in table 1 and compared with the computation times attained by Bouvier and Messoumian on the IBM 7044. The inflexible backtracking algorithm was more efficient than the flexible backtracking algorithm in the case of the Bouvier-Messoumian problem. There may be several reasons for this, differences in computer programming being one of them. The flexible backtracking algorithm is more sophisticated since there is flexibility in choosing which variable to fix at the alternative value ; however, this also complicates the book keeping work in the computer program, and so might make the algorithm slower. Presumably, the flexible algorithm could result in better choices of paths in the tree of partial solutions and so in some cases might be a better method to use. The inflexible backtracking algorithm allowed variables to be fixed at either zero or one in forward steps while the flexible backtracking algorithm fixed variables first to one. For this reason, the inflexible algorithm would find small values of the criterion function more quickly than the other method, so that truncation procedures would be more effective. This suggests that perhaps the flexible backtracking algorithm should be altered to allow both values of zero or one to be chosen in forward steps.

For the inflexible backtracking algorithm, in most cases each problem was solved in almost the same time regardless which choice procedure was used. This is because the rounding procedure, while not choosing the variable which gives the least increase in the criterion function, chooses a variable which increases the criterion function relatively little and requires less computation and tableau search time then the least increase choice procedure. Naturally, which choice procedure is more efficient depends on the nature of the problem being solved.

A mixed integer problem due to Driebeek [5] with 27 constraints and 40 variables, 9 of which are zero-one, was solved. The tableau for this problem is represented by table 2. (L denotes a « less than or equal to » constraint and E denotes an « equality » constraint.) The problem deals with four factories which ship to eight demanders. The first 14 constraints have to do with the capacities of the factories. For instance, the first three constraints say that

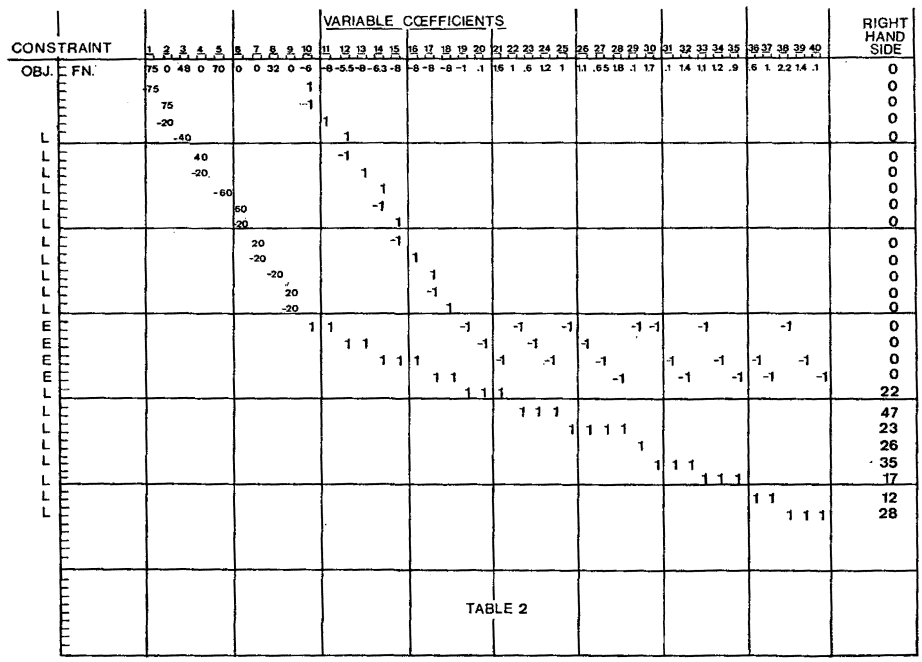
$$\begin{aligned} x_{10} &\leqslant 75x_1 \\ 75x_2 &\leqslant x_{10} \\ x_{11} &\leqslant 20x_2 \end{aligned}$$

In other words, we have a factory which can produce 75 units. If the factory produces anything, then $x_1 = 1$. If nothing is produced, then $x_1 = 0$. If more than 75 units of production is desired, then the factory may produce up to twenty more units. In this case $x_2 = 1$,

which also forces $x_{10} = 75$. Then $x_{10} + x_{11}$ gives the total output of the factory. There are similar constraints for the other three factories. The next set of constraints deals with how the factories' production is allocated among the demanders, and the last set of constraints gives the demands. The problem is to minimize total costs minus total revenues,

TABLE 1. — *Computation results*
CDC 6500

Problem	Problem size 0-1 Variable × Constraints	Flexible Backtracking time (s)	Inflexible backtracking time (s)		Bouvier and Messoumian time (s) IBM 7044
			Rounding choice procedure	Least increase choice procedure	
No. 16	20 × 20		15.03	15.3	126
No. 21	23 × 20	68.9	48.5	45.5	216
No. 22	25 × 20		21.5	18.8	196
No. 23	27 × 20		50.9	84.1	426
No. 24	28 × 20	144.2	61.3	65.2	912
No. 25	30 × 20	405.2	134.4	141.5	> 960 (No solution reported)



i.e. to maximize profits. The solution for maximum profit of 1130.350 was obtained and verified after 8 seconds on the CDC 6500.

A similar mixed integer problem due to Wilson [12] was also solved. This problem has 77 constraints and 80 variables, 16 of which are required to be zero or one. This problem was solved in 13.5 seconds on the CDC 6500.

BIBLIOGRAPHY

1. Egon BALAS, « An additive algorithm for solving linear programs with zero-one variables », *Operations Research*, **13**, No. 4 (1965).
2. Egon BALAS, « Discrete programming by the filter method », *Operations Research*, **15**, No. 5 (1967).
3. B. BOUVIER and G. MESSOUMIAN, « Programmes linéaires en variables bivalentes, algorithme de Balas », Université de Grenoble, France, juin 1965.
4. Norman J. DRIEBEEK, « An algorithm for the solution of mixed integer programming problems », *Management Science*, **12**, 576-587 (1966).
5. Norman J. DRIEBEEK, Unpublished Problems.
6. F. GLOVER, « A multiphase-dual algorithm for the zero-one integer programming problem », *Operations Research*, **13**, No. 6, 879-919 (1965).
7. R. E. GOMORY, « Outline of an algorithm for integer solutions to linear programs », *Bull. Am. Math. Soc.*, **64**, 275-278 (1958).
8. G. W. GRAVES and A. B. WHINSTON, « An algorithm for the quadratic assignment problem », in J. Abadie editor, integer and nonlinear programming, north holland pub. 1970.
9. A. H. LAND and A. G. DOIG, « An automatic method of solving discrete programming problems », *Econometrica*, **28**, No. 3 (1960).
10. NGHIEM Ph. Tuan, « A flexible tree search method for integer programming problems », Krannert Institute Paper No. 232, Purdue University (1968).
11. Harvey WAGNER, « The dual simplex algorithm for bounded variables », *Nav. Res. Log. Qu.*, **5**, 257-61 (1958).
12. R. C. WILSON, Unpublished Problems, Esso Research Center.