A BRANCH-AND-BOUND ALGORITHM FOR ZERO-ONE MIXED INTEGER PROGRAMMING PROBLEMS

Ronald E. Davis

Stanford University, Stanford, California

David A. Kendrick

University of Texas, Austin, Texas

and

Martin Weitzman

Yale University, New Haven, Connecticut

(Received August 7, 1969)

This paper presents the results of experimentation on the development of an efficient branch-and-bound algorithm for the solution of zero-one linear mixed integer programming problems. An implicit enumeration is employed using bounds that are obtained from the fractional variables in the associated linear programming problem. The principal mathematical result used in obtaining these bounds is the piecewise linear convexity of the criterion function with respect to changes of a single variable in the interval [0, 1]. A comparison with the computational experience obtained with several other algorithms on a number of problems is included.

MANY IMPORTANT practical problems of optimization in management, economics, and engineering can be posed as so-called 'zero-one mixed integer problems,' i.e., as linear programming problems in which a subset of the variables is constrained to take on only the values zero or one. When indivisibilities, economies of scale, or combinatoric constraints are present, formulation in the mixed-integer mode seems natural. Such problems arise frequently in the contexts of industrial scheduling, investment planning, and regional location, but they are by no means limited to these areas.

Unfortunately, at the present time the performance of most comprehensive algorithms on this class of problems has been disappointing. This study was undertaken in hopes of devising a more satisfactory approach. In this effort we have drawn on the computational experience of, and the concepts employed in, the LAND AND DOIG[16] Healy,[13] and DRIEBEEK[15] algorithms. Similar approaches are contained in the work of BEALE

AND SMALL.[2] A related enumeration scheme is contained in GRAVES AND WHINSTON.[10]

Our approach is basically a branch-and-bound method of enumeration. While this is not generally the most 'glamorous' type of algorithm, our experience indicates that it works well in solving practical problems. As with any branch-and-bound algorithm, efficiency derives primarily from choosing branches and bounds in a manner that is computationally effective.

We proceed with a discussion of the algorithm, followed by a presentation of our computational results.

1. GENERAL THEORY

LET

$$y=\begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix}$$

be an n-vector of zero-one variables, each of which is constrained to be zero or one. Let

$$x=\begin{bmatrix} x_1 \\ \vdots \\ x_m \end{bmatrix}$$

be an m-vector of continuous nonnegative variables. The standard form of the zero-one mixed integer programming problem is the following:

$$\text{minimize } z=\phi(y, x)=cy+dx, \tag{1}$$

subject to

$$A\begin{bmatrix} y \\ x \end{bmatrix}\geqq b \quad \text{for} \quad \begin{bmatrix} y \\ x \end{bmatrix}\geqq 0, \tag{2}$$

$$y_i\leqq 1 \quad \text{for} \quad i=1, \cdots, n, \tag{3}$$

$$y_i=0 \text{ or } 1 \quad \text{for} \quad i=1, \cdots, n. \tag{4}$$

There are 2^n possible y vectors satisfying (4). Some of these may admit of no x satisfying (2) and hence are infeasible.

Consider the solution $(\bar{y}, \bar{x})$ to the problem (1), (2), (3). [We call $(\bar{y}, \bar{x})$ the solution to the 'unconstrained' problem because the integer constraints (4) are missing.] If $\bar{y}$ is already a vector of zeros and ones, the problem is solved. Far more likely, it is not. In that case, we seek more information about the local properties of the unconstrained solution.

Consider the variable y_1. Let $z_1(\tilde{y}_1)$ be the optimal value of the objective function defined parametrically in terms of $\tilde{y}_1$ for the problem (1), (2), (3) with the additional constraint $y_1=\tilde{y}_1$ for $\tilde{y}_1\epsilon[0, 1]$. It can be shown that $z_1(\tilde{y}_1)$ is a piecewise linear convex function of $\tilde{y}_1$ (cf. DANTZIG,[3] p. 168).

To map out the entire function $z_1(\tilde{y}_1)$ would require, in general, the solution of many linear programming problems. We settle instead for the more easily attainable behavior of the two linear segments to the direct right and left of $\bar{y}_1$. The slopes of these two segments can be read out of the linear programming tableau. (Strictly speaking, the assertion is true only if the basis is nondegenerate. If degeneracy is involved, we may be able to obtain only a lower bound on the slope of each segment, difficult to picture geometrically, but otherwise the results are not altered.)

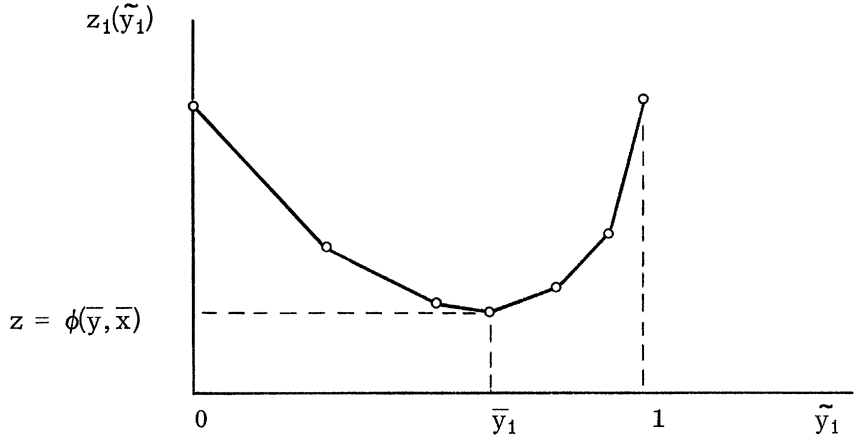

Fig. 1. A piecewise linear convex function.

The slopes are obtained as follows. Suppose, first of all, that y_1 is in the optimal basis of the associated linear program and that $0<\bar{y}_1<1$. Let $\{w_j : j \epsilon NB\}$ be the set of nonbasic variables, let

$$y_1 + \sum_{j \epsilon NB} \bar{a}_{ij} w_j = \bar{y}_1$$

be the row in the updated simplex tableau corresponding to y_1, and let

$$z + \sum_{j \epsilon NB} \bar{c}_j w_j = \varphi(\bar{y}, \bar{x})$$

be the reduced-cost row (where $\bar{c}_j \geqq 0$ for $j \epsilon NB$). Along the extreme rays adjacent to $(\bar{y}, \bar{x})$ generated by increasing one w_j and holding the other nonbasic variables fixed, we have:

(i) If $a_{ij}>0$, increasing w_j by $1/\bar{a}_{ij}$ will decrease y_1 by 1 and increase the objective function by $\bar{c}_j/\bar{a}_{ij}$.

(ii) If $\bar{a}_{ij}<0$, increasing w_j by $1/-\bar{a}_{ij}$ will increase y_1 by 1 and the objective function by $\bar{c}_j/-\bar{a}_{ij}$.

(iii) If $\bar{a}_{ij}=0$, increasing w_j has no effect on y_1.

Since the feasible region is contained in the convex cone generated by

these rays (cf. DANTZIG,[3] p. 155), it follows that

$$m_0 = \min_{\bar{a}_{ij}>0, j\in NB} \bar{c}_j/\bar{a}_{ij} \quad \text{and} \quad m_1 = \min_{-\bar{a}_{ij}>0, j\in NB} \bar{c}_j/-\bar{a}_{ij}, \quad (5)$$

(where m_0 is defined to be infinite if all $\bar{a}_{ij} \leqq 0$, and m_1 is defined to be infinite if all $\bar{a}_{ij} \geqq 0$) are lower bounds (which are exact except possibly in the case of degeneracy) on the absolute value of the slopes of $z_1(\tilde{y}_1)$ at $\bar{y}_1$.

If y_1 is a nonbasic variable at level 0, $m_1 = \bar{c}_1$; if it is a basic variable at level 0 or 1, only m_1 (or m_0) need be computed from (5).

From the convexity of $z_1(\tilde{y}_1)$ it follows that $[q_0(j), q_1(j)]$ given by

$$\begin{aligned} q_0(j) &\equiv z_1(\bar{y}_1) + \bar{y}_1 \cdot m_0 \leqq z_1(0), \\ q_1(j) &\equiv z_1(\bar{y}_1) + (1-\bar{y}_1) \cdot m_1 \leqq z_1(1) \end{aligned} \quad (6)$$

are lower bounds on the objective function value if y_j is forced to the values 0 or 1, respectively (cf. DRIEBEEK[6]). Bounds for $y_2, \cdots, y_n$ can be computed analogously. These bounds are used repeatedly in the algorithm for branching decision, and are rapidly computed directly from quantities in the updated simplex tableau provided by the simplex algorithm.

2. BRANCH-AND-BOUND SEARCH

EXPOSITIONS AND EXAMPLES of branch-and-bound-search algorithms abound in the literature (e.g., LAND AND DOIG,[16] BALAS,[1] AND GEOFFRION[8]); we reiterate here only that such an algorithm proceeds by sequentially partitioning the potential solution space into disjoint subsets determined by specifying the values of some of the discrete variables (called a *'partial assignment'* or 'partial solution'); that a lower bound is computed for the value of the objective on all elements of each such subset (the set of *completions* of a partial assignment); and that the partitioning process can be represented diagramatically in the form of a tree (where *nodes* correspond to partial assignments).

The relative efficiency of a branch-and-bound algorithm over total enumeration derives from the ability to eliminate from consideration large subsets of potential optimal solutions. Two principal ways of eliminating such subsets are: (i) to show that no feasible completion of a given partial assignment exists; and (ii) to show that no optimal completion of a given partial assignment exists. Balas and others following him (*see* FLEISCHMANN,[7] IVANESCU AND RUDEANU[14]) have developed a set of additive feasibility tests for these purposes that utilize the zero-one restriction on the discrete variables in an essential way. Land and Doig[16] suggested performing subset elimination by referring to the solutions to linear programs obtained by fixing the variables in a partial assignment to their assigned values. Although developed in isolation from one another, these

two approaches are clearly not mutually exclusive; incorporation of both into the same algorithm may yield further improvements in computational efficiency (*see* Geoffrion[9]). The algorithm reported here, however, results from an investigation of the ways in which the Land and Doig approach can be enhanced by using tables of bounds derived from those described above.

Thus, if $I=\{1, 2, \cdots, n\}$ is the index set of the zero-one variables, and if I_0 and I_1 are the index sets of variables in a partial assignment assigned to zero and one, respectively, let $LP(I_0, I_1)$ be the associated 'restricted' linear program (with variables indexed by I_0 and I_1 fixed at their assigned values and let $Q(I_0, I_1)$ be the associated bounds matrix defined by (6). Then if I_f is the set of *free* variables ($I_f=I-[I_0 U I_1]$),

$$\Psi(I_0, I_1) \equiv \max_{j \epsilon I_f} \min [q_0(j), q_1(j)] \tag{7}$$

is a lower bound on the value of the objective function for any completion of (I_0, I_1). All completions of (I_0, I_1) can thus be eliminated if (i) LP (I_0, I_1) is infeasible, or (ii) $\Psi(I_0, I_1) \geqq \bar{z}$, where $\bar{z}$ is an upper bound on the optimal solution value. If neither of these conditions obtains, but $q_u(j) \geqq \bar{z}$ for some $j \epsilon I_f$, $u=0$ or 1, then all completions of (I_0, I_1) yielding an improved solution value must satisfy $y_j=(1-u)$. In this case, (I_0, I_1) may be enlarged to (I_0', I_1'), where

$$I_0'=I_0,\ I_1'=I_1 \cup \{j\} \tag{8a}$$

if $u=0$, or

$$I_0'=I_0 \cup \{j\},\ I_1'=I_1 \tag{8b}$$

if $u=1$; if y_j is fractional valued in the solution to $LP(I_0, I_1)$, the solution to $LP(I_0', I_1')$ yields an improved set of bounds $Q(I_0', I_1')$ to which the above tests may be applied again. Augmenting partial assignments in this way can substantially reduce computation times.

Some virtually costless improvements on the bounds matrix $Q(I_0, I_1)$ follow from the observation that, since any completion of (I_0, I_1) is also a completion of (J_0, J_1) if $J_0 \subseteq I_0$ and $J_1 \subseteq I_1$,

$$\tilde{Q}(I_0, I_1)=\sup_{J_0 \subseteq I_0, J_1 \subseteq I_1} Q(J_0, J_1) \tag{9}$$

is another valid, and generally improved, bounds matrix. A more readily computable improvement over $Q(I_0, I_1)$, which lends itself well to the algorithm given below, is provided by the following updating procedure:

$$\begin{aligned} Q(I_0 \cup \{j\}, I_1) &\leftarrow \sup \{Q(I_0 \cup \{j\}, I_1), Q(I_0, I_1)\}, \\ Q(I_0, I_1 \cup \{j\}) &\leftarrow \sup \{Q(I_0, I_1 \cup \{j\}), Q(I_0, I_1)\}. \end{aligned} \tag{10}$$

The results reported below were obtained using bounds matrices updated in this way.

To obtain large lower bounds and avoid redundant assignments of variables connected by constraints, partitioning of the completions of a partial assignment (I_0, I_1) that cannot be enlarged or eliminated is accomplished by choosing that fractional valued variable y_j in the solution to $LP(I_0, I_1)$ that has maximum zero or one bound, and considering the two sets of completions of (I_0, I_1) determined by assigning y_j to 0 or y_j to 1.

3. STATEMENT AND DISCUSSION OF THE ALGORITHM

LET $\bar{z}$ be the current upper bound on the optimal solution value, and S be the set of partial assignments that remain to be considered at iteration ν. S_0 is defined to consist of the vacuous assignment with both I_0 and I_1 empty (which is the 'specified' node for iteration 1). Then the νth iteration of the algorithm proceeds as follows:

A. *Node Evaluation*

Solve $LP(I_0, I_1)$ for the currently specified assignment (I_0, I_1) and, if feasible, compute $Q(I_0, I_1)$ and $\psi(I_0, I_1)$, using (6), (10), and then (7).

(1) If $LP(I_0, I_1)$ is infeasible or $\psi(I_0, I_1) \geqq \bar{z}$, form S_ν by removing (I_0, I_1) from $S_{\nu-1}$ and go to step C.

(2) If y satisfies (4), record the feasible solution as the best yet found, set $\bar{z}$ to the current objective function value, form S_ν by removing (I_0, I_1) from $S_{\nu-1}$, and go to step C.

(3) If $q_0(j) \geqq \bar{z}$ or $q_1(j) \geqq \bar{z}$ for any free variable y_j, $j \epsilon I_f$, enlarge (I_0, I_1) according to (8), and, if any such y_j is fractional valued in the solution to $LP(I_0, I_1)$, repeat step A. Otherwise, continue with step B.

B. *Node Formation* (or branching)

For the current (possibly enlarged) assignment (I_0, I_1), find $j^* \epsilon I_f$ such that y_{j^*} is the fractional-valued variable in the solution to $LP(I_0, I_1)$ with the largest zero or one bound, and form S_ν by replacing (I_0, I_1) in $S_{\nu-1}$ with the two assignments $(I_0 U \{j^*\}, I_1)$ and $(I_0, I_1 U \{j^*\})$. If $q_u(j^*) < q_{1-u}(j^*)$ $(u = 0 \text{ or } 1)$, a lower bound on the completions with $y_{j^*} = u$ is $\psi(I_0, I_1)$, and, with $y_{j^*} = 1 - u$, is $q_{1-u}(j^*)$.

C. *Node Selection*

Choose the next partial assignment to be evaluated from S_ν according to one of the following rules:

Option (1). If (I_0, I_1) was eliminated or yields an integer solution, use option (2). Otherwise specify the assignment determined in B where y_{j^*} is assigned to the value with the smaller bound.

Option (2). Specify the node in S_ν with the least lower bound. If S_ν is empty or none with lower bound less than $\bar{z}$ exist, the search is complete.

In option (1), the current assignment is partially completed at successive iterations until it becomes infeasible, is bounded off by the current upper bound $\bar{z}$, or produces an improved feasible solution. A disadvantage of this approach is that many more nodes may be evaluated than are necessary to prove convergence. One advantage is that storage require-

ments are minimal. Option (2) is the 'flooding' strategy of always evaluating the partial assignment with lowest lower bound. The advantage of this procedure is that only those assignments that are necessary to prove convergence are evaluated. A disadvantage is that large storage capacity may be required. Use of random-access disk-removed computer memory is a limiting factor for us, so this approach was usually used.

The efficacy of the tests in A depends on obtaining a good upper bound $\bar{z}$ early in the search. One of the most successful approaches tried was to run the algorithm with option (1) and $\bar{z}=+\infty$ in the very beginning. This usually resulted in a near-optimal feasible integer solution; then option (2) could be applied.

4. EXPERIENCE WITH THE ALGORITHM

TABLE I gives a resumé of computational experience with our algorithm (*DKW*) using updated bounds matrices and the 'flooding' strategy search. The table also gives comparisons with the experience of others where possible. Option 2 was used for all the runs of our algorithm reported here.

Solution time in the table refers to the time required on an IBM 7094 to obtain the optimum mixed integer solution, including the solution to the unconstrained problem. In the solution of the Kendrick problem with the Healy algorithm, all but four of the variables could have been bounded off after 24 minutes using an upper bound obtained from other solutions of the problem. Though such a bound is not available in the Healy algorithm, it is used here for comparative purposes. It is assumed that the enumeration of the remaining 16 lattice points would have required 10 minutes. The same type of adjustment is made for the number of *LP*'s and iterations. In solving the same problem with Driebeek's algorithm, the run was halted after enumerating 100 of the 128 'unbounded' lattice points in 44 minutes. Enumeration of the remaining 28 lattice points is assumed to have required 12 minutes. The same type of adjustment is made for the number of *LP*'s and iterations.

ACKNOWLEDGMENTS

THIS RESEARCH was supported in part by the National Science Foundation and in part by the Agency for International Development and Harvard University. We are grateful to the referees for referring us to the related works of BEALE AND SMALL and of GRAVES AND WHINSTON.

Of the programs we used, the Land and Doig algorithm was coded by PAUL ROBERTS and BOB BURNS, the Healy algorithm was coded by KENDRICK AND WEITZMAN, and the algorithm described in this paper was coded

TABLE I

COMPUTATIONAL EXPERIENCE WITH THE ALGORITHM

Problem name	Problem number	Algorithm	Number of integer variables	Total number of variables	Total number of constraints	Matrix entries	Solution time			
							Minutes	Number of *LP's*	Number of iterations	Pivot Steps
Kendrick[15] 11 I.V.	1		11	412	116	1767				
		Healy					37.	58	876	
		Driebeek					59.	130	NA	
		DKW					10.3	36	433	653
Markowitz-Manne[17]	2		21	51	33	285				
		DKW					5.	62	411	
Davis[4] 100 I.V.	3		100	317	197	1261				
		DKW					5.5	16	189	277
Balas 1[1]	4		5	16	14	58				
		DKW					0.1	3	8	8
		Balas						3		
Balas 2[1]	5		10	29	22	119				
		DKW					0.2	6		
		Balas						5		
Balas 3[1]	6		9	25	19	99				
		DKW					0.1	3	11	NA
Balas 4[1]	7		12	34	25	143				
		DKW					0.4	15	82	96
		Balas						39		
IBM Test Problem 9[11,12]	8		15	68	56	253				
		DKW						128	657	805
		LIP1						69		952

by DAVIS (*see* Davis[5]). We are indebted to JUN ONAKA for able programming assistance in solving the problems referred to in Table I.

REFERENCES

1. E. BALAS, "An Additive Algorithm for Solving Linear Programs with Zero-One Variables," *Opns. Res.* **13,** 517–546 (1965).
2. H. M. L. BEALE AND R. E. SMALL, "Mixed Integer Programming by a Branch-and-Bound Technique," *Proceedings of the IFIP Congress, 1965*, pp. 450–451, W. A. KALENICH (ed.) Spartan Press, Washington, D. C. and MacMillan, London. *See also* E. M. L. BEALE *Mathematical Programming in Practice*, Sir Isaac Pitman and Sons, London, 1968.
3. GEORGE B. DANTZIG, *Linear Programming and Extensions*, Princeton University Press, Princeton, N. J., 1963.
4. RONALD E. DAVIS, "A Hospital Scheduling Program: Formulation and Solution in Zero-One Variables," B.A. Thesis, Committee on Applied Mathematics, Harvard University, 1967.
5. ———, "Program Description for MFOR-(0-1) MIP: A Code for Zero-One Mixed Integer Programming Problems," Report No. 71, Project for Quantitative Research in Economic Development, Harvard University, Cambridge, Mass., September, 1967.
6. NORMAN J. DRIEBECK, "An Algorithm for the Solution of Mixed Integer Programming Problems," *Management Sci.* **12,** 576–587 (1966).
7. BERNHARD FLEISCHMANN, "Computational Experience with the Algorithm of Balas," *Opns. Res.* **15,** 153–155 (1967).
8. A. M. GEOFFRION, "An Improved Implicit Enumeration Approach for Integer Programming," Rand Corporation Memorandum RM-5644-PR, June 1968.
9. ———, "An Improved Implicit Enumeration Approach for Integer Programming," *Opns. Res.* **17,** 437–454 (1969).
10. G. GRAVES AND A. WHINSTON, "A New Approach to Discrete Mathematical Programming," *Management Sci.* **15,** 177–190 (1968).
11. JOHN HALDI, "Twenty-Five Integer Programming Test Problems," Working Paper #43, Graduate School of Business, Stanford University, December 1964.
12. ——— AND LEONARD M. ISAACSON, "A Computer Code for Integer Solutions to Linear Programs," *Opns. Res.* **13,** 946–959 (1965).
13. W. C. HEALY, JR., "Multiple Choice Programming," *Opns. Res.* **12,** 122–138 (1964).
14. P. L. IVANESCU AND S. RUDEANU, *Boolean Methods in Operations Research*, Springer-Verlag, Berlin, 1968.
15. DAVID A. KENDRICK, *Programming Investment in the Process Industries*, M.I.T. Press, Cambridge, Mass., 1967.
16. A. H. LAND AND A. G. DOIG, "An Automatic Method of Solving Discrete Programming Problems," *Econometrica* **28,** 397–520 (1960).
17. H. M. MARKOWITZ AND A. S. MANNE, "On the Solution of Discrete Programming Problems," *Econometrica* **25,** 84–98 (1957).