Operations Research

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

Letter to the Editor—Computational Results of an Integer Programming Algorithm

Patrick D. Krolak,

Letters to the Editor

COMPUTATIONAL RESULTS OF AN INTEGER PROGRAMMING ALGORITHM

Patrick D. Krolak

Vanderbilt University, Nashville, Tennessee

(Received October 4, 1967)

This note reports on a method for solving the general integer linear programming problem that is called the Bounded Variable Algorithm. It first describes the basic algorithm and then makes a comparison of limited scope between the Bounded Variable Algorithm and other published algorithms on a set of common problems.

IN THE FIELD of integer linear programming, which has been rapidly expanding in recent years, the work has proceeded along several lines: The best known line is probably the cutting-plane approach,[1-4] but modifications of dynamic programming have been tried by others,[5-7] and recent papers have dealt with truncated enumeration.

The work in truncated enumeration has centered around formulations of the integer linear programming (ILP) problem as a (0–1) ILP, i.e., reducing all general variables to the sums of binary variables,[8-13] but a few papers have attacked the problem directly.[14-16] In addition to these exact algorithms, there have been several heuristics proposed.[17-21]

This note gives the computational experience with an algorithm called the "Bounded Variable Algorithm," which uses truncated enumeration to solve general ILP problems without recourse to binary variables.

THEORY

WITHOUT TOO much of a loss in generality (for details on how to take care of the remaining cases, *see* reference 21), we can state the general ILP problem as: maximize $z=\boldsymbol{cx}$, subject to $A\boldsymbol{x}\leqq \boldsymbol{b}$ and $\boldsymbol{D}\leqq\boldsymbol{x}\leqq\boldsymbol{U}$, where $\boldsymbol{U}$, $\boldsymbol{D}$, $\boldsymbol{c}$, and $\boldsymbol{x}$ are integer n-vectors, A is an $m\times n$ matrix, and $\boldsymbol{b}$ is an m-vector.

Now, suppose we solve the following linear programming problems subject to the same constraints as the ILP except for the integer requirement in $\boldsymbol{x}$:

$$\max \boldsymbol{cx}, \tag{1}$$

$$\max \textstyle\sum_B x_i, \tag{2}$$

$$\max \textstyle\sum_B c_i x_i, \tag{3}$$

$$\min \textstyle\sum_B c_i x_i, \tag{4}$$

$$\min \boldsymbol{cx}, \tag{5}$$

where B is the set of indices of variables in the optimal basis of (1). The above five results give us five additional constraints on the integer solution that we shall, for obvious reasons, call redundant constraints (for example, x_I^*, the integer optimal solution, must satisfy $\boldsymbol{c}x_I^* \leqq \max\,[\boldsymbol{c}x]$, where $[a]$ is the greatest integer $\leqq a$).

Append the five new redundant constraints to the A-matrix and the $\boldsymbol{b}$-vector and drop all the constraints in the original A-matrix that are not binding at optimality. Call the appended matrix A_1 and the matrix formed by the dropped constraints A_2. Now reorder the variables in such a fashion that all the variables whose indices are in B are first in sequence.

The Bounded Variable Algorithm is based on a lemma that depends upon a simple observation. Given two n-vectors $\boldsymbol{D}$ and $\boldsymbol{U}$, where $\boldsymbol{D} \leqq \boldsymbol{U}$, and an $\boldsymbol{A}$ and a $\boldsymbol{b}$, then we shall define two integer n-vectors $\boldsymbol{D}'$ and $\boldsymbol{U}'$ where the jth component of $\boldsymbol{U}'$ satisfies

$$U_j' = \min\left\{ \min_{\text{for all } I \ni a_{IJ} > 0} \left\{ \left[\frac{b_I - \sum_{k \epsilon J-} a_{Ik} U_k - \sum_{k \epsilon J+} a_{Ik} D_k}{a_{IJ}} \right] \right\}, [U_J] \right\}$$

and the jth component of $\boldsymbol{D}'$ satisfies

$$D_j' = \max\left\{ \max_{\text{for all } I \ni a_{IJ} < 0} \left\{ \left[\frac{b_I - \sum_{k \epsilon J-} a_{Ik} U_k - \sum_{k \epsilon J+} a_{Ik} D_k}{a_{IJ}} + 1 \right] \right\}, [D_J] \right\}$$

where $J-$ is the set of all indices with $k \neq J$ and having $a_{Ik} \leqq 0$, and $J+$ is the set of all indices with $k \neq J$ and having $a_{Ik} > 0$.

Lemma. *For $\boldsymbol{D}'$ and $\boldsymbol{U}'$ as defined above, $\boldsymbol{D}' \leqq \boldsymbol{x} \leqq \boldsymbol{U}'$ for all-integer $\boldsymbol{x}$ satisfying $A\boldsymbol{x} \leqq \boldsymbol{b}$ and $\boldsymbol{D} \leqq \boldsymbol{x} \leqq \boldsymbol{U}$.*

A trivial observation is that there exists no integer $\boldsymbol{x}$ satisfying the constraint set if $\boldsymbol{D}'$ is not less than or equal to $\boldsymbol{U}'$.

Now, using the lemma, the redundant constraints, and the reordered variables, we state the Bounded Variable Algorithm:

(1) Given two integer vectors $\boldsymbol{U}$ and $\boldsymbol{D}$ that are upper and lower bounds on any possible solution vector $\boldsymbol{x}$, we define a list L_i to be a collection of integers

$$L_i = \{j \mid j = D_i, D_i + 1, \cdots, U_i\}.$$

(2) Pick an integer from list L_1, say k_1. Set $x_1 = k_1$.

(3) Use the lemma to get a new upper and lower bound on x_2, given that we have set the bounds on x_1 to be equal to k_1. If $D_2' > U_2'$, go to step (5). If $D_2' \leqq U_2'$ generate a list L_2 and pick a value from L_2 to set x_2 at and go to (4).

(4) Continue as in step (3), generating a list for each variable in succession and setting the appropriate variable equal to some member of the list. At the ith variable, the information that the variables $x_1, \cdots, x_{i-1}$ have been set at some possible value within their feasible limits is used to calculate the ith variable's bounds. Eventually either we get to the nth list, where we successfully apply the lemma to both the A_1 and A_2 matrices and hence a feasible solution is found, or some variable is found, say x_j, whose bounds are $D_J' > U_J'$, in which case, according to the theorem, we can conclude that no possible solution exists having the values assumed for $x_1, \cdots, x_{J-1}$. If a feasible solution is generated, go to (7). If the bounds are contradictory, go to (5).

(5) In attempting to get bounds in the Jth variable, it was found that no integer solution was possible. Go to the $(J-1)$st list and remove from it the value at which x_{J-1} was assigned. Either list L_{J-1} is empty or it is not. If it is not, assign x_{J-1} a value from the remaining members of L_{J-1} and go to (4). If the list L_{J-1} is empty, then go to (6).

(6) If $J>2$, then there can be no solution having the assigned values $x_1, \cdots, x_{J-2}$, since this combination has been tried for all possible values of x_{J-1} and has

TABLE I

COMPARISON OF COMPUTATIONAL EXPERIENCE ON COOK AND ECHOLS DATA

	n	m	Z_I^*	Z_{LP}^*	B.V.A. 7044 min	COOK 7072 min	7072 min RAO exact	DREBES heuristic 360-50 min	7072 min RAO heuristic	ECHOLS heuristic 7072 min
(C)	20	8	1204	1271	1.52	5.30		1	0.1 (R)	(F)
(C)	20	9	1327	1516	2.05	9.83		1	0.1 (R)	(F)
(C)	20	10	1264	1345	3.94	9.51		1	0.1 (R)	(F)
(C)	20	10	4616	4791	3.66	18.66		2	0.1 (R)	(F)
(C)	25	8	3221	3332	0.65	9.90		1	0.2 (R)	(S)
(C)	25	8	5367	5454	7.60	19.95		3	0.3 (R)	(F)
(C)	25	9	1774	1898	8.95	18.20		2	0.1 (R)	(F)
(C)	20	10	23544	23858	11.90	62.08		4	0.8 (R)	(S)
(C)	24	15	5002	5247	9.70	39.47		3	0.2 (R)	(F)
(C)	21	21	5153	5334	16.00	57.17		4 (F)	0.2 (R)	(F)
(E)	9	7	107097	107118	0.70	—		4		4.431 (F)
(E)	21	27	540	596	8.60	—		2		1.936 (F)
(E)	12	10	17	18	0.26	—	83.00	0.5		0.628

(C) Data taken from the appendix of Cook's report.
(E) Data taken from the appendix of Echol's report.
(R) Rao's suboptimal heuristic produces answers of the order 0.9 Z*.
(S) An optimal answer reported but no computing time.
(F) A suboptimal answer reported.

failed to generate a feasible solution. Set $J=J-1$ and go to (5). If $J\leqq 2$, then no possible solution exists. Go to (8).

(7) A feasible solution has been found. (Note that at the point where the feasible solution was found there were bounds on the nth variable such that $D_n'\leqq U_n'$, and any choice from the nth list would produce a feasible x). We now set x_n to the largest number of list L_n if $c_n\geqq 0$ or to the smallest member if $c_n<0$. This gives us the largest feasible solution for the previously set values of $x_1, \cdots, x_{n-1}$. Update the redundant constraints and store the feasible solution. Now go to (5).

(8) If no feasible solutions are found, then there is no integer solution. If feasible solutions are generated, then the last one generated is optimal.

This procedure is, of course, finite, and is similar to several proposals involving truncated enumeration.[13,15,16] The bounding device is, of course, extremely simple and further steps listed in references 21 and 22 are needed to make the program computationally efficient.

TABLE II

Comparison of Computational Experience on Haldi and IBM Data

Source	m	n	IBM 7090 sec. IPM3[(a)]	IBM 7090 sec. LIPI[(b)]	CDC 3600 sec. ILP-2-1[(c)]	CDC 3600 sec. ILP-2-2[(d)]	CDC 3600 sec. IPSC[(e)]	IBM 7094 sec. BALAS[(f)]	IBM 360-65 -7044 sec. B.V.A.	IBM 7044 sec. GEOFFRION[(g)]
Haldi 5	5	6	F	9.0	F	F	79.9	1	30	—
6	5	6	F	7.5	F	3.2	43.48	12	12	—
7	5	4	F	7.8	F	F	F	1	24	
8	5	4	F	6.4	F	3.0	F	13	9	2
9	6	6	5.18	3.2	F	3.59	5.48	—	8	3
10	12	10	71.1	9.1	F	F	F	—	17	4
IBM 1	7	7	2.3	1.86	1.01	1.14	1.04	—	18	1
2	7	7	2.8	3.0	1.05	1.08	1.14	—	21	1
3	4	3	2.63	2.86	0.75	0.62	0.48	—	4	1
4	15	15	5.93	11.66	3.5	3.08	3.64	400	23	6
5	15	15	51.6	66.5	F	26	62.8	>600	154	114
9	50	15	633	473	F	75	95.4	—	—	36
4 pt. problem	6	8	2.2	1.767	0.89	0.90	0.76	—	3	—

F, Failed to converge after 14,000 iterations.

(a) IPM, R. Levitan and Gomory of IBM, Share distribution #1190.[3]

(b) LIPI, Haldi and Issacson of Standard Oil.[3]

(c) ILP2-1 and ILP2-2, Summers of CDC.[3]

(d) IPSC, Woolsey of Sandia Corporation.[3]

(e) Balas (0–1), Lemke and Spielberg of IBM.[24]

(f) Geoffrion, A. M. Geoffrion of Rand Corporation.[12]

COMPUTATIONAL EXPERIENCE

The computational experience of this algorithm is listed in three tables. It will be noted by those familiar with the works of the other authors that the data cover an extremely diverse set of problem types going from low to high ranges in density, per cent negativity, and bounds on variables.[22] The data in Table II are made up of problems that have very small ranges on the variables, and hence the binary codes can be expected to be very efficient; still, the Bounded Variable Algorithm does solve the problems in reasonable times. In addition to the work reported in the tables, the author has solved over 50 additional problems and found that, for problems of 50 variables or less, the code has so far always produced a close, if not optimal, solution in 10 minutes of IBM 7044 time.

ACKNOWLEDGMENTS

THE AUTHOR would like to thank the computing facilities of the Applied Mathematics Department of Monsanto Company and Southern Illinois University for the necessary computer time. Discussions with L. COOPER of Washington University, St. Louis, Missouri, also were of great help in preparing this paper. The work was done at the Southern Illinois University, Edwardsville, Illinois, and at Washington University, St. Louis, Missouri. The work was supported in part by a contract with the Atomic Energy Commission.

TABLE III

n	m	7044, min
P 50	5	13
K 48	6	9
H 15	15	3
H 15	15	2

P Average time of three problems, the original stated in PETERSEN,[11] the other two having the b vector changed. The variables are all (0–1).

K Average time of ten problems. Five additional problems were terminated after 12 minutes with feasible solutions close to the LP optimal solution.

H Average time of 6 problems of HILLIER's type I.[15]

H Average time of 6 problems of HILLIER's type II.[15]

REFERENCES

1. R. E. GOMORY, "All-Integer Integer Programming Algorithm," IBM Research Report RC-189, January, 1960.
2. R. D. YOUNG, "A Primal (All Integer) Interger Programming Algorithm: Antecedents, Description, Proof of Finiteness, Exemplification," Work Paper No. 52, Graduate School of Business, Stanford School of Business, Stanford University, December, 1964.
3. C. A. TRAUTH AND R. E. WOOLSEY, "Practical Aspects of Integer Linear Programming," Sandia Corporation Research Report, SC-R-66-925, August 1966.
4. M. L. BALINSKI, "Integer Programming: Methods, Uses, Computation," *Management Sci.* **12**, 253–313 (1965).
5. H. GLASS, "The Application of Dynamic Programming to the Solution of the Zero-One Integer Linear Programming," Doctoral Dissertation, Sever Institute of Technology, Washington University, January, 1966.

6. V. Rao, "Some Approaches to Integer Programming," Doctoral Dissertation, Sever Institute of Technology, Washington University, June, 1967.
7. G. A. Gorry and J. F. Shapiro, "An Adaptive Group Theoretic Algorithm for Integer Programming Problems," Technical Report No. 39, Operations Research Center, Massachusetts Institute of Technology, May, 1968.
8. E. Balas, "An Additive Algorithm for Solving Linear Programs with Zero-One Variable," *Opns. Res.* **13,** 517–546 (1965).
9. R. J. Freeman, "Computational Experience with the Balas Integer Programming Algorithm," *Opns. Res.* **14,** 935–940 (1966).
10. Bernard Fleischman, "Computational Experience with the Algorithm of Balas," *Opns. Res.* **15,** 153–154 (1967).
11. C. C. Petersen, "Computational Experience with Variants of the Balas Algorithm Applied to Selection of R & D Projects," *Management Sci.* **13,** 736–750 (1967).
12. A. M. Geoffrion, "An Improved Implicit Enumeration Approach for Integer Programming," Memorandum of the Rand Corp., RM-5644-PR, June, 1968.
13. F. Glover, "Truncated Enumeration Methods for Solving Pure and Mixed Integer Linear Programs," Working Paper, Operations Research Center, University of California, Berkeley, May, 1966.
14. A. H. Land and A. G. Doig, "An Automatic Method of Solving Discrete Programming Problems," *Econometrica* **28,** 497–520 (1960).
15. F. S. Hillier, "An Optimal Bound-and-Scan Algorithm for Integer Linear Programming," Technical Report No. 3, Department of Industrial Engineering, Stanford University.
16. R. Cook, "An Algorithm for Integer Linear Programming," Doctoral Dissertation, Sever Institute of Technology, Washington University, January, 1966.
17. F. S. Hillier, "Efficient Suboptimal Algorithms for Integer Linear Programming with an Interior," Technical Report No. 2, Department of Industrial Engineering, Stanford University.
18. R. E. Echols, "The Solution of Integer Linear Programming Problems by Direct Search," *J. ACM*, 75–84 (January, 1968).
19. C. Drebes and L. Cooper, "Investigations in Integer Linear Programming by Direct Search Methods," Report No. COO-1493-10, Department of Applied Mathematics and Computer Science, Washington University, 1967.
20. S. Reiter and D. Rice, "Discrete Optimizing Solution Procedures for Linear and Nonlinear Programming Problems," Paper No. 109, Institute for Research in the Behavioral, Economic and Management Sciences, Purdue University, May, 1965.
21. P. D. Krolak, "The Bounded Variable Algorithm for Solving Integer Programming Problems," AEC Research Report No. COO-1493-18, Department of Applied Mathematics and Computer Science, Washington University, September, 1967.
22. ———, "An Improved Bounded Variable Algorithm with Adaptations to the Special Cases of the Generalized Knapsack Problem and the Integer Fixed Charge Problem," Working Paper No. 1, Southern Illinois University, Business Division, Edwardsville, September, 1968.

23. ———, "Problems Currently Used to Test Integer Linear Programming Algorithms," Technical Report, Information Engineering Dept., Vanderbilt Univ., May 1969.

24. C. E. LEMKE AND K. SPIELBERG, "Direct Search Zero-One and Mixed Integer Programming," Technical Report No. 3, IBM Corporation, New York, September, 1967.

AN APPROACH TO SOME STRUCTURED LINEAR PROGRAMMING PROBLEMS

John M. Bennett and David R. Green

University of Sydney, Sydney, Australia

(Received October 10, 1966)

A recent paper by J. M. BENNETT describes a decomposition algorithm for the class of linear programming problems commonly called 'angular systems.' This note draws attention to a variant of the algorithm that is particularly suited to problems in which certain submatrices are sparse.

WE CONSIDER the decomposition algorithm for the class of linear programming problems, commonly called 'angular systems,' that was presented in a recent paper by J. M. Bennett,[1] and assume that its submatrices $\boldsymbol{B}^i$ are sparse. The result is a variant of the algorithm that is particularly suited to problems where this assumption holds.

This variant involves modifying the calculation of the shadow costs. The method outlined in Bennett's paper [equation (7)] uses a matrix

$$\boldsymbol{G}=[\cdots|\boldsymbol{B}^{i(1)}-\boldsymbol{B}^{i(2)}\boldsymbol{A}^{i(2)I}\boldsymbol{A}^{i(1)}|\cdots] \tag{1}$$

for computing the shadow costs. This matrix is used for no other purpose in the algorithm.

An alternative procedure for computing the shadow costs is to calculate the last row of the transforming matrix $\boldsymbol{T}$ and then to post-multiply this row by

$$\begin{bmatrix} \boldsymbol{A}^{1(1)} & & & \\ & \ddots & & \\ & & & \boldsymbol{A}^{k(1)} \\ \boldsymbol{B}^{1(1)} & \cdots & & \boldsymbol{B}^{k(1)} \end{bmatrix} \tag{2}$$

The last P elements of the last row of $\boldsymbol{T}$ are given by $\boldsymbol{F}_{P.}$, and the remaining elements are available as

$$-[\boldsymbol{F}_{P.}\boldsymbol{B}^{1(2)}\boldsymbol{A}^{1(2)I}|\cdots|\boldsymbol{F}_{P.}\boldsymbol{B}^{k(2)}\boldsymbol{A}^{k(2)I}] \tag{3}$$