MANAGEMENT SCIENCE
Vol. 24, No. 3, November 1977
Printed in U.S.A.

A MODIFIED BENDERS' PARTITIONING ALGORITHM FOR MIXED INTEGER PROGRAMMING*

DALE MCDANIEL† AND MIKE DEVINE‡

As applied to mixed-integer programming, Benders' original work made two primary contributions: (1) development of a "pure integer" problem (Problem P) that is equivalent to the original mixed-integer problem, and (2) a relaxation algorithm for solving Problem P that works iteratively on an LP problem and a "pure integer" problem. In this paper a modified algorithm for solving Problem P is proposed, in which the solution of a sequence of integer programs is replaced by the solution of a sequence of linear programs plus some (hopefully few) integer programs. The modified algorithm will still allow for taking advantage of any special structures (e.g. an LP subproblem that is a "network problem") just as in Benders' original algorithm. The modified Benders' algorithm is explained and limited computational results are given.

In 1962, Benders [2] proposed a partitioning approach for solving programming problems that involve a mixture of either different types of variables or different types of functions. Two common applications are the solving of mixed integer linear programming problems and mixed linear and nonlinear problems. As applied to mixed integer problems, Benders' approach (1) defined a "pure" integer problem that is equivalent to the original problem, and (2) devised an iterative relaxation scheme for solving the "pure" integer problem. One drawback to this method is that it required solving a "pure" integer problem at each iteration. The purpose of this paper is to present an alternative relaxation scheme for solving the "pure" integer problem. Some limited computational results are also given.

1. Problem Statement and Notation

The general mixed integer problem may be stated as:

$$\begin{aligned} &\text{Minimize: } X_0 = CX + C'Y \\ &\text{Subject to: } AX + A'Y \geqslant B \\ &\qquad\qquad DY \geqslant B' \\ &\qquad\qquad X \geqslant 0 \quad Y \geqslant 0 \text{ and integer,} \end{aligned} \tag{P1}$$

where A is $m \times n$, A' is $m \times p$, D is $m' \times p$ and the vectors are appropriately dimensioned. The constraints $DY \geqslant B'$ are those (if any) that do not contain any continuous variables.

If one lets the integer variables Y assume some values, say Y', the dual of the remaining LP problem is

$$\begin{aligned} &\text{Maximize: } U_0 = U(B - A'Y') \\ &\text{Subject to: } UA \leqslant C \\ &\qquad\qquad U \geqslant 0. \end{aligned} \tag{P2}$$

To this problem, let us add a constraint $UE \leqslant M$, where E is an $m \times 1$ vector of 1's and M is some appropriately large number.[1] We will call this new problem P2′. If P2

* Accepted by Arthur M. Geoffrion; received April 30, 1974. This paper has been with the authors 9 months, for 2 revisions.

† California State University at Northridge.
‡ University of Oklahoma.

[1] This is equivalent to adding a new continuous variable to P1 with a coefficient of M in the objective function and coefficients of 1 in each constraint.

is bounded, then the added constraint is redundant, so that P2 and P2′ are equivalent. If for a given value of Y, the optimal solution to P2 is unbounded, then the optimal solution to P2′ will be bounded, with the constraint $UE \leqslant M$ "tight". Thus, if we solve P2′ and $UE \leqslant M$ is tight, then we can conclude that P2 is unbounded. Note that the feasible region of P2′ (and P2) is independent of Y, so that regardless of what value Y may assume, the optimal solution of P2′ will be a vertex of the feasible region. Let us denote each of the vertices of this space by $U^p, p = 1, \ldots, T$.

2. Benders' Partitioning Algorithm

Benders derived the following "pure"[2] integer problem which is equivalent to P1.

$$\begin{aligned} &\text{Minimize: } Z \\ &\text{Subject to: } Z \geqslant C'Y + U^p(B - A'Y), \qquad p = 1, \ldots, T, \\ &\qquad DY \geqslant B' \\ &\qquad\quad Y \geqslant 0 \text{ and integer.} \end{aligned} \tag{P3}$$

Note that this problem has T constraints (one for each of the vertices of P2′) in addition to the m' constraints of $DY \geqslant B'$. As T is usually very large, however, Benders proposed a relaxation scheme to solve the problem. The procedure suggested by Benders is one in which P3 is relaxed by starting with a small subset of the T constraints in P3 (usually only 1 constraint). Then one successively generates constraints for the "pure" problem by alternately solving P2′ and the relaxed "pure" problem. (See, for example, Garfinkel and Nemhauser [6] for a more detailed development of the algorithm.) Very briefly, assuming P1 is feasible and bounded, Benders' algorithm is

Step 0. Initialization: $t = 1$, $B_u = +\infty$ and select some ϵ (convergence criteria). Select some U^1 that is feasible for P2′.

Step 1. Solve the relaxed "pure" integer problem:

$$\begin{aligned} &\text{Minimize: } Z \\ &\text{Subject to: } Z \geqslant C'Y + U^i(B - A'Y), \qquad i = 1, \ldots, t, \\ &\qquad DY \geqslant B' \\ &\qquad\quad Y \geqslant 0 \text{ and integer.} \end{aligned} \tag{P4}$$

Let Z^t and Y^t be the solution. If Z is unbounded from below, take Y^t to be some value that gives Z^t some arbitrarily large negative value. (P4 is the relaxed version of P3.)

Step 2. Generate the most violated constraint of P3 by solving the linear program P2′:

$$\begin{aligned} \text{Maximize: } & U_0 = U(B - A'Y^t) \\ \text{Subject to: } & UA \leqslant C \\ & UE \leqslant M \\ & U \geqslant 0. \end{aligned}$$

Let the solution to this linear program be U_0^{t+1} and U^{t+1}.

Step 3. Check convergence criteria: $B_u \leftarrow \min\{B_u, U_0^{t+1} + C'Y^t\}$. If $Z^t > B_u - \epsilon$,

[2] The quote around pure is used to signify that P3 actually contains one continuous variable. Several special methods for solving this problem have been proposed (see, for example, Balas [1] and Zoutendijk [12]).

stop—the optimal solution has been reached. Otherwise, add the constraint $Z \geqslant C'Y + U^{t+1}(B - A'Y)$ to P4. Let $t = t + 1$. Return to Step 1.

At the end of any iteration t, Z^t represents a lower bound on the optimal solution to P1, and it is monotonically increasing. Also, $C'Y^t + U_0^{t+1}$ represents an upper bound, but it is not necessarily monotonically decreasing. Therefore, the best upper bound (B_u) is given by $B_u = \min\{C'Y^i + U_0^{i+1}\}$, $i = 1, \ldots, t$.

There have been some encouraging results reported (for example, see [4], [5], [7] and [11]) which show that Benders' algorithm can often solve the mixed integer program in relatively few iterations. Unfortunately, when the original problem involves many integer variables, solving the "pure" integer problem at each iteration can be very costly.

3. A Modified Algorithm

The purpose of the iterative process of Benders' algorithm is to successively generate constraints for P4. This is accomplished by solving P4 to optimality and then solving P2′ in order to generate the most violated constraint of P3. Of course, any U^p ($p = 1, \ldots, T$), or in fact any feasible U, not necessarily an extreme point, forms a legitimate constraint that can be added to P4. The objective of the proposed algorithm is to generate "good" constraints for P4 without having to solve the "pure" integer problem at each iteration.

The approach of this alternative algorithm is to solve the "pure" integer subproblem P4 for some number of iterations as if it were a linear program, i.e., the integer restrictions are dropped. The resulting solution to P4 (which may be fractional) is then used in P2′ to generate a new constraint for P4. The constraints generated for P4 are still legitimate since any extreme point from P2′, regardless of how it was determined, may form a constraint for P4. The hope is that many of the necessary constraints for P4 may be generated by solving linear programs in the place of integer programs.

There are a variety of heuristic rules possible for determining when to solve P4 as an IP or when to solve it as an LP. Some possibilities are (a) Continue LP iterations on P4 until no further iterations are possible (in which case one has found the LP solution to P1 via Benders' partitioning), and from then on solve P4 as an IP; (b) solve P4 as an LP for the first k iterations, say $k = 10$, and then switch to IP's; (c) if $B_u - Z^t \leqslant \delta$, then switch to the IP.

One should note that when P2 is solved with a fractional value for Y^t, then $U_0^t + C'Y^t$ is not necessarily an upper bound for P1 since the Y's are not integer feasible. Thus, with rule (c) above one is not comparing upper and lower bounds on P1, but rather on the LP version of P1 (i.e., P1 with the integer restriction relaxed).

An algorithm using a combination of these heuristics is given below:

Step 0. Same as Benders' original algorithm. Also, specify k and δ ($\geqslant \epsilon$).

Step 1. If $t > k$, go to Step 1A. If $t \leqslant k$, solve the linear program:

$$\begin{aligned} &\text{Minimize: } Z \\ &\text{Subject to: } Z \geqslant C'Y + U^i(B - A'Y), \qquad i = 1, \ldots, t, \\ &\qquad DY \geqslant B' \\ &\qquad Y \geqslant 0. \end{aligned}$$

Let Y^t and Z^t be the solution. Go to Step 2. (Note no integer restriction on Y.)

Step 1A. Solve the integer program:

$$\begin{aligned} &\text{Minimize: } Z \\ &\text{Subject to: } Z \geqslant C'Y + U^i(B - A'Y), \qquad i = 1, \ldots, t, \\ &\qquad DY \geqslant B' \\ &\qquad Y \geqslant 0 \text{ and integer.} \end{aligned}$$

Let Y^t and Z^t be the solution. Go to Step 2.

Step 2. Same as Benders' original algorithm.

Step 3. $B_u \leftarrow \min\{B_u, U_0^{t+1} + C'Y^t\}$. If $Z^t > B_u - \epsilon$, and Step 2 was entered from Step 1A, stop—optimal solution has been reached.

If $Z^t > B_u - \delta$, and Step 2 was entered from Step 1, set $k = 0$ and go to Step 1A. If $Z < B_u - \delta$, add the constraint $Z \geqslant C'Y + U^{t+1}(B - A'Y)$ to the subproblem of Step 1 and Step 1A. Let $t \leftarrow t + 1$. Go to Step 1.

4. Computational Results

Some preliminary computational tests have been obtained on the algorithm presented and compared to Benders' original algorithm. Table 1 summarizes the characteristics of some of the problems that have been solved. (Another set of very small test problems were solved in the process of debugging the code, but these results are not presented here due to space limitations. The omission of these small problems does not affect the conclusions drawn.) The number of continuous variables given is exclusive of slack or artificial variables and all integer variables are 0-1.

TABLE 1

Problem	Number of Continuous Variables	Number of 0-1 Variables	Number of Rows	Type
3a–3f	36	24	14	Production Allocation with Set-up Costs
4a–4f	36	24	38	Production Allocation with Piecewise Linear Costs
5a–5f	12	25	12	Capital Budgeting
6c–6e	24	24	32	Fixed Charge Transportation

It is not feasible in this paper to describe fully each of these test problems. However, complete descriptions (including all numerical data) can be found in [10]. Very briefly, problems 3a–3f are production allocation problems with set-up costs, which can be modeled as:

$$\text{minimize} \quad \sum_{i=1}^{n} \sum_{k=1}^{l} \sum_{j=1}^{m} C_{kij} X_{kij} + \sum_{k=1}^{l} \sum_{i=1}^{n} v_{ki} Y_{ki}$$

$$\text{subject to:} \quad \sum_{i=1}^{n} X_{kij} \geqslant d_{kj}, \qquad k = 1, \ldots, l; j = 1, \ldots, m$$

$$\sum_{k=1}^{l} \sum_{j=1}^{m} t_{ki} X_{kij} + \sum_{k=1}^{l} h_{ki} Y_{ki} \leqslant r_i, \qquad i = 1, \ldots, n$$

$$\sum_{j=1}^{m} X_{kij} \leqslant M Y_{ki}, \qquad k = 1, \ldots, l; \qquad i = 1, \ldots, n,$$

$$\text{all } X_{kij} \geqslant 0 \text{ and all } Y_{ki} = 0, 1,$$

where the decision variables are

X_{kij} = amount of production of product k at plant i for delivery to warehouse j;

Y_{ki} = 1 if product k is produced at plant i, 0 otherwise;

and the problem data are

C_{kij} = per unit cost of producing product k at plant i (excluding set-up cost) and delivering to warehouse j,

ν_{ki} = set-up cost for product k at plant i,

d_{kj} = demand at warehouse j for product k,

t_{ki} = production time for product k at plant i (excluding set-up),

h_{ki} = set-up time for product k at plant i,

r_i = total available production time at plant i,

M = sufficiently large number.

Problems 4a–4f are similar production allocation problems but with piecewise linear concave production costs.

Problems 5a-5f are simple capital budgeting problems of the following form:

$$\text{maximize } \sum_{i=1}^{n} C_i X_i + \sum_{j=1}^{m} C_j' Y_j$$

$$\text{subject to: } \sum_{i=1}^{n} A_{ij} X_i + Y_j - Y_{j-1} = b_j, \qquad j = 1, \ldots, m,$$

$$\text{all } X_i = 0, 1;\ Y_0 = 0, \text{ and all } Y_j \geqslant 0,$$

where the decision variables are

$$X_i = 1 \quad \text{if project } i \text{ is selected,}$$
$$= 0 \quad \text{otherwise;}$$
$$Y_j = \text{amount of cash left uninvested at the end of period } j;$$

and the problem data are

C_i = present value of project i,

C_j' = present rate of loss on money left uninvested in period j,

A_{ij} = amount of investment required in period j for project i,

b_j = budgeted amount of cash for all projects in period j.

Problems 6c–6e are fixed-charge transportation problems of the type described in [8], [9]. In fact, problems 6d and 6e correspond to problems 2 and 3 respectively of [8, p. 68].

Table 2 lists the timings and the number of iterations required for both of the algorithms on each test problem. For the modified algorithm, the number of iterations required was separated into those where only LP subproblems were solved and those where an IP solution was required. All problems were run on the UNIVAC 1108 located on the Madison Campus of the University of Wisconsin and all timings are in CPU seconds. Relatively unsophisticated and inefficient codes were used to solve the LP subproblem and the IP subproblems, and no attempt was made to take advantage of special problem structures; e.g., the transportation subproblems in problems 6c–6e were solved with the standard simplex method. For all of these problems $\epsilon = \max\{0.001, 0.0003Z^t\}$, $\delta = 0.06Z^t$, and k = arbitrarily large number.

Of interest in these results is the fact that the modified algorithm resulted in at most 50% more iterations and in two cases fewer iterations than Benders' algorithm. Since one is trading linear programming problems for integer programming problems at most of the iterations, this is an encouraging result. The inability of Benders' algorithm to find answers in five minutes on a large number of the problems, caused largely by the computation times to solve the IP problems, restricts many of the comparisons.

Probably the most interesting and promising result is that on all of the problems solved to optimality with the modified algorithm, it was necessary to solve only one IP

TABLE 2
Computational Results

	Benders' Original Algorithm		Modified Algorithm		
				No. of Iterations	
Problem	Time	No. of Iterations	Time	LP	IP
3a	4.16	7	5.53	8	1
3b	5.23	9	8.13	15	1
3c	4.00	7	5.20	8	1
3d	5.23	11	5.90	10	1
3e	3.31	7	4.57	8	1
3f	5.15	7	7.82	10	1
4a	290.55*	2 +	83.32	17	1
4b	290.25*	3 +	89.57	18	1
4c	272.82	10	66.79	11	1
4d	209.27	10	97.76	14	1
4e	290.25*	2 +	101.97	19	1
4f	290.85*	5 +	85.78	17	1
5a	126.00	5	63.11	4	1
5b	13.12	3	11.79	3	1
5c	98.47	4	64.05	2	1
5d	20.32	3	19.05	2	1
5e	29.73	4	18.23	3	1
5f	7.71	4	9.31	5	1
6c	288.75*	17 +	76.88	18	1
6d	288.00*	20 +	74.62	17	1
6e	191.56	15	289.20*	19	1 +

* No final solution in stated time.

problem. That is, after P4 was solved as an IP for the first time, the procedure terminated with the optimal solution. (This result also holds for the smaller test problems that are not presented here.)

To get a better comparison between Benders' original algorithm and the modified algorithm, Table 3 gives some computational results for some of the same problems using a slightly improved computer code.[3] This table gives a breakdown of CPU time between the LP and IP problems, and times are CPU seconds for an IBM 360/50.[4] For the results in Table 3, $\epsilon = \max\{0.001, 0.003Z'\}$, $\delta = 0.05Z'$, and $k =$ arbitrarily large number. Notice in comparing the results in Tables 2 and 3 that in most cases there was a different number of iterations required. This resulted because (1) different convergence tolerances were used; (2) different amounts of round-off error were caused by different algorithms for the LP problems and different computers, and (3) in the case of alternative optimums in the subproblems, it is likely that the different codes found different solutions, thereby affecting the subsequent sequence of solutions. The results of Tables 2 and 3 generally agree in that the modified algorithm was superior in CPU time; but in cases where Benders' algorithm finished, it is superior in terms of the number of iterations. One interesting difference in the two sets of results is that problem 6d required more than three IP iterations in Table 3. This demonstrates that one cannot always expect only one IP iteration with the modified

[3] The authors wish to gratefully acknowledge the efforts of David H. Cookerly and Hee Man Bae in developing this improved computer code. The code was partially developed as part of [3], and most of the results in Table 3 also appear in [3].

[4] As a rough estimate the UNIVAC 1108 used for the results in Table 2 is about five times faster than the IBM 360/50.

TABLE 3

Comparison of Original Benders' and New Algorithm

	Benders's Algorithm				Modified Algorithm				
		CPU Time			Iteration		CPU Time		
Prob.	Iteration	Total	LP	IP	LP	IP	Total	LP	IP
4a	5 +	291.6*	19.6	272.0	15	1	168	45.9	122.1
4b	9	255.2	38.5	216.7	17	1	112.0	47.1	64.9
4c	8	227.3	43.8	183.5	19	1	179.4	66.7	112.7
4d	5 +	291.1*	38.3	252.8	20	1	288.1	70.1	218.0
4e	10 +	291.7*	56.4	235.3	18	1	207.9	63.5	144.4
4f	5 +	291.8*	32.0	259.8	23	1	177.7	84.4	93.3
5a	4	434.2	6.5	427.7	4	1	13.0	9.2	3.8
5c	3	197.0	6.2	190.8	4	1	9.8	8.9	0.9
6d	6 +	291.8*	23.2	268.6	17	3	224.9**	9.8	215.1

* No solution in stated time.

** These times for problem 6d using the new algorithm are for a IBM 370/155J computer.

algorithm; in general, it will depend on the particular problem, the convergence criteria and the computer code.

Figures 1, 2 and 3 plot the convergence rates for the original Benders' algorithm and the modified algorithm for problems 4b, 4c and 6d respectively. The plots indicate where IP solutions were first determined by the modified algorithm. As one would expect, the original Benders' algorithm converges more rapidly in terms of the number of iterations.

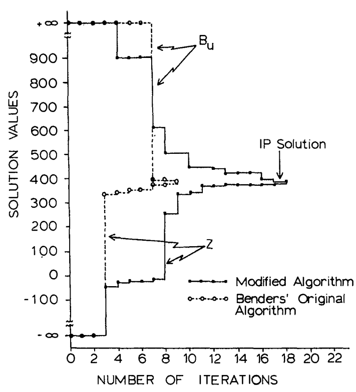

FIGURE 1. Convergence Rates for Problem 4b.

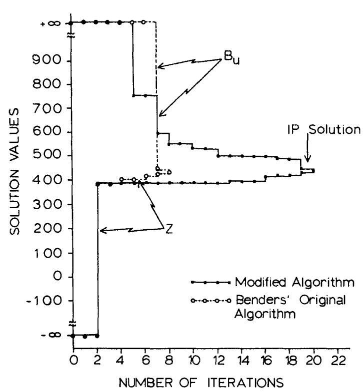

FIGURE 2. Convergence Rates for Problem 4c.

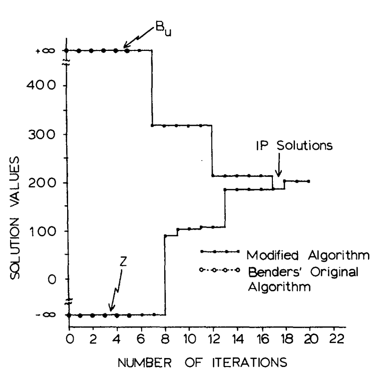

FIGURE 3. Convergence Rates for Problem 6d.

In summary, the computational tests are on relatively small problems using crude computer codes, and therefore no definite conclusions can be drawn about the possible advantages of the modified Benders' algorithm. However, thus far the results for the modified algorithm appear promising. To draw firmer conclusions, more experimentation will be required using more sophisticated codes on larger problems to determine which of the Benders' approaches is superior, and more importantly, whether any of the partitioning procedures are comparable to branch and bound techniques.

References

1. Balas, E., "Minimix and Duality for Linear and Nonlinear Mixed-Integer Programming," in *Integer and Nonlinear Programming*, J. Abadie, Ed., American Elsevier (1970), pp. 385–418.
2. Benders, J. F., "Partitioning Procedures for Solving Mixed-Variables Programming Problems," *Numerische Mathematic*, Vol. 4 (1962), pp. 238–252.
3. Cookerly, David H., "Computational Aspects of Various Solution Techniques for the Pure Integer Subproblem Associated with Benders' Partitioning Algorithm," unpublished Master's Thesis, University of Oklahoma (1974).
4. Etschmaier, M. M. and Richardson, R. J., "Improving the Efficiency of Benders' Decomposition Algorithm for a Special Problem in Transportation," Technical Report No. 14, Department of Industrial Engineering, University of Pittsburg (May 1973).
5. Florian, M., Guerin, G. and Bushell, G., "The Engine Scheduling Problem in a Railway Network —Part 1," Publication No. 89, Department of Information, University of Montreal (May 1972).
6. Garfinkel, R. S. and Nemhauser, G. L., *Integer Programming*, John Wiley and Sons, New York, 1972.
7. Geoffrion, A. M. and Graves, G. W., "Multicommodity Distribution System Design by Benders' Decomposition," *Management Science*, Vol. 20, No. 5 (January 1974), pp. 822–844.
8. Gray, P., "Mixed Integer Programming Algorithms for Site Selection and Other Fixed-Charge Problems Having Capacity Constraints," Ph.D. Dissertation, Stanford University, Stanford, California (November 1967); issued as a technical report by Stanford Research Institute (November 1967).
9. ———, "Exact Solution of the Fixed-Charge Transportation Problem," *Operations Research*, Vol. 19, No. 6 (October 1971), pp. 1529–1538.
10. McDaniel, Roger Dale, "Computational Aspects of Various Benders-Based Procedures for Mixed Integer Programming," unpublished Ph.D. Dissertation, University of Oklahoma (1973).
11. Muckstadt, J. A. and Wilson, R. C., "An Application of Mixed-Integer Programming Duality to Scheduling Thermal Generating Systems," *IEEE Transactions*, Vol. PAS-87, No. 12 (Dec. 1968), pp. 1968–1978.
12. Zoutendijk, G., "Enumeration Algorithms for the Pure and Mixed Integer Programming Problem," *Proceedings of the Princeton Symposium on Mathematical Programming*, H. W. Kuhn, Ed., Princeton University Press (1970).