M. M. Guignard
K. Spielberg

Mixed-integer Algorithms for the (0,1) Knapsack Problem

Abstract: An enumerative scheme is presented for the (0,1) knapsack problem as a specialization of the state enumeration method. Techniques are explored for rendering search procedures more efficient by systematic use of information generated during execution of the algorithm. The inequalities of Benders and Gomory-Johnson are exploited to yield implicit enumeration tests in the special case of the knapsack problem. In a comparative study of eight algorithms and of the utility of certain approximations and inequalities, computational results are given for twelve knapsack problems, each having ten (0,1) variables. The effectiveness of these enumerative algorithms are thus tested in a relatively simple framework.

Introduction

The knapsack problem is concerned with filling some allocated space, e.g., a knapsack, of volume r with various items k (taken once only, or not at all) of volume b_k and having associated "profit" p_k selected from a set of p items, so as to maximize the overall profit:

$$\max \sum p_k t_k = z\,,$$
$$\sum b_k t_k + s = r \qquad s \geq 0\,, \quad t_k \in \{0\,,1\}\,,$$
$$k \in K = (1\,,2\,,\cdots,p)\,. \qquad (1)$$

The optimal solution will not, in general, completely fill the space r but will leave a "slack" portion s unfilled.

We shall assume that all b_k and p_k are positive and that the ratios p_k/b_k are in decreasing order from left to right:

$$p_k/b_k \geq p_{k+1}/b_{k+1}\,, \quad k = p-1\,, \quad p-2\,,\cdots,1\,. \qquad (2)$$

The knapsack problem is an important zero-one integer or mixed-integer problem, both in its own right and as a possible aid in the solution of general mixed zero-one problems. We give in this paper an enumeration procedure which may be viewed as a specialization of the state enumeration method [1 , 2]. However, the particular nature of the problem suggests a number of deviations and modifications of the enumerative scheme, e.g., the use of three states derived from easily obtained approximating solutions, and the incorporation of inequalities from the Gomory asymptotic problem.

Our primary objective is to demonstrate how well suited is the state enumeration technique, or a variant of it, for a problem of special structure. We believe that elements of the method could be incorporated advantageously in other algorithms for the knapsack problem (see, e.g., Refs. 3 and 4). We suggest also that other special problems which can be solved readily as linear programs and can be altered easily to give integer-feasible solutions, may be particularly amenable to solution by techniques such as those advocated here. The set covering problem and the plant location problem are two examples [5 , 6].

A secondary objective is to explore methods for rendering search procedures more efficient by the systematic use of information generated during execution of the algorithm. In particular we have exploited to this end the inequalities of Benders and the asymptotic problem of Gomory. For the all-integer (0,1) knapsack problem, the Gomory-Johnson inequalities [7 , 8], which are in general over all variables, including the slacks, can be transformed to inequalities over the zero-one variables only and can then be used in exactly the same manner as the Benders inequalities. Hence, we were able to test the effectiveness of these new ideas within a relatively simple framework.

We can thus use the zero-one inequalities to yield efficient implicit enumeration tests, leading to the fixing of variables or to the establishment of infeasibility [1 , 2 , 9].

For the mixed (0,1) knapsack problem and for general mixed (0,1) problems, one would have to use somewhat more complicated devices, e.g., linear programming over the inequalities. For such methods see Ref. 8. The actual computational results of this paper are for the pure (0,1) case.

This paper first points out some of the differences between branch-and-bound and enumerative programming and then shows how modifications of the state enumeration method can be applied to the solution of the given knapsack problem. Various forms of heuristic devices are next discussed, including the use of the linear programming solution, of three different integer solutions, and of Benders and Gomory-Johnson inequalities. Numerical results for these various algorithms are then compared with branch-and-bound techniques for the solution of twelve knapsack problems.

Branch-and-bound and enumeration procedures

There is a good deal of confusion in the literature as to the difference between (implicit) enumeration and branch-and-bound (BB) programming, especially "single branch" BB programming, in which one branch of the search tree is pursued to the end, with backtracking, before another is taken up. The reader may wish to consult Ref. 9 on such matters, e.g., on the terminology (a fairly obvious one) of single-branch and multibranch trees. Reference 9 should also be consulted for further references to the basic papers on BB and enumerative programming.

The essential difference lies in the nature of the problems that are actually solved: *relaxed problems* (i.e., problems in which the constraints $t_k \in \{0, 1\}$ are relaxed to $0 \leq t_k \leq 1$ in the case of BB programming) and *constrained problems*, in which specific integer values are substituted for all or a part of the t_k, in the case of enumeration.

This distinction can become somewhat blurred by the fact that an enumeration algorithm may also employ relaxed auxiliary problems intermittently. In the enumeration scheme of this paper, relaxed auxiliary problems are always used, even when all t_k are (0, 1) variables. In strict enumeration one could avoid all linear programs but usually only at the expense of losing valuable information.

In the state enumeration algorithm [1, 2], an auxiliary linear program may suggest (e.g., by rounding) a "state", which is a determination as to which of the "free" variables ($k \in F$)—the set of variables t_k which have not been explicitly fixed at 0 or 1—are to be set to 1 ($k \in F1$), 0($k \in F2$), or be left free ($k \in F3$) in the solution of certain basic constrained "state problems."

The manner in which F is partitioned into $F1, F2, F3$, determines the order in which the enumeration proceeds and the inequalities that are generated. The arbitrariness available in the choice of the state furnishes the essential flexibility of the state enumeration method. Of course, it may not be easy to assign a state (to partition) intelligently. See Refs. 1 and 2 for details. The usefulness of a flexible procedure was, we believe, sufficiently demonstrated by the computational results of Ref. 5.

It ought to be emphasized that, in general, linear programming need not necessarily be used. Linear programming can be used intermittently only, e.g., only at the start. The state can be defined by means of some other method. Moreover, even the state problems need not be solved at all nodes, but only at the "terminal" nodes at which all zero-one variables have been fixed. In contrast to this, BB programming is intrinsically based on the resolution of two linear programs (or, more generally, two relaxed programs) at each node of the search tree.

For the knapsack problem the situation is quite special, in that any linear programming solution of (1), with some variables possibly fixed at 0($k \in Z$) and some at 1($k \in E$), can be made to have the property that no more than one component in the solution, say t_f, has a fractional value.

A BB algorithm, then, will always round variable t_f to 0 for one branch and to 1 for the other. It will then proceed from one of the pending nodes in the same fashion. The computational results in this paper will always refer to this true "multibranch" BB method with, in general, many pending nodes, as opposed to the single-branch approach.

The more interesting approach for us, however, is that of state enumeration, modified so as to take into account the special nature of the problem. For such an enumeration algorithm the inequalities become the motive force. Since the inequalities are almost solely responsible for the curtailment of the search process, the manner in which they are generated (e.g., by correct choices of state) becomes a factor of overriding importance.

State enumeration for the knapsack problem

Consider problem (1) as given initially, i.e., with all variables free:

$$F = K, \quad E = \varnothing, \quad Z = \varnothing. \tag{3}$$

All variables are free, i.e., none are fixed at 0 or 1. No distinction has yet been made among the free variables. In general, the state enumeration algorithm has the following raw structure:

1) Define l, the "level", to be (in a sense) the distance from the origin of the search, i.e., let it be the number of variables that have been fixed explicitly in the last sequence of "forward steps" from the origin (level 0). Initially, the level is set to 0. It is increased in forward steps, decreased in "backups".

2) If there are free variables left, go to 3). Otherwise, solve a terminal state problem, i.e., solve the original problem with the integer variables fixed as specified by E and Z. Go to 4).

3) *Forward step*
a) Determine a state, i.e., partition F into the three sets $F1, F2, F3$. In other words partition K:

$$K = E + Z + F1 + F2 + F3. \quad (4)$$

This state can be determined trivially, from the previous state, or by means of auxiliary computations. Pick a branch variable k^* in F.
b) Increase l by one.
Set $t_{k^*} = 1$ if k^* is in $F2$ or $F3$, $t_{k^*} = 0$ if k^* is in $F1$; i.e., adjoin k^* to E if it was in $F2$ or $F3$ and adjoin k^* to Z if it was in $F1$. Of course, modify $F1$, $F2$, $F3$ correspondingly. Store k^* for later use in backups. Go to 2).

4) *Backup*
Retrieve the value of k^* which led to the current node (level).
Complement the variable, i.e., if t_{k^*} is currently 1(0) fix it now at 0(1).
Reduce l by one. If $l = -1$, terminate. Otherwise, go to 2).

This description, of course, lacks motivation and all those details and additional features which are necessary to make the procedure effective, such as:

- In 3a) one must decide on a method of determining the state. Frequently, one may decide to solve a non-terminal-state problem, obtained from the original problem by substitution of $t_k = 1$ for k in $F1$ and $t_k = 0$ for k in $F2$, t_k being left to vary between 0 and 1 for k in $F3$. In the initial phases of the computation one would, of course, like to get a solution to the original problem. In general, however, computational evidence indicates that, wherever possible, it is desirable to choose the state close to a hypothesized good solution [5].
- When a state problem or an auxiliary problem has been solved, one can always obtain a Benders inequality [9, 10] over the integer variables. From the auxiliary problem one usually is able to obtain Gomory-Johnson inequalities [7, 8]. All of those are necessary conditions for obtaining integer solutions. The Benders inequalities may involve z^* (a given or already computed bound on the objective function) in which case they are conditions for obtaining integer solutions with objective functions better than, or equal to, z^*.

  These inequalities can be utilized systematically in a search for the possibility of fixing ("cancelling") variables at the current level l. On a backup to a previous level, all cancellations at the current level must be rescinded.
- In the general case, the inequalities may afford the best way of choosing k^*. One will choose k^* such that the forward step alters the state so that the new state problem at the successor state differs from the previous one.

For details, the reader is referred to Ref. 2 or 8, although we realize that those descriptions may be too concise to be sufficiently clear. It is in fact difficult to define any general rules for selecting effective computational features. It is up to the analyst to decide on such matters in relationship to the particular problem at hand. In this paper we have tried to do this to some extent for the knapsack problem.

Approaches to solving the knapsack problem

The linear programming solution for (1) is obtained rather easily. We therefore made the natural decision to solve it, as an auxiliary problem, in every iteration. From the linear programming solution we then construct heuristically three different solutions, if possible, to the original problem. With each such integer solution we may, if we wish, associate a state and also a corresponding linear programming solution and tableau. We shall give details presently, including examples.

From each linear programming (LP) tableau we may, by inspection, deduce a linear inequality after Benders. (The Benders inequalities derived from the integer solutions are only integral multiples of the original constraints and are of no interest here). Also, we can in general derive an asymptotic problem after Gomory and construct a Gomory-Johnson inequality over the t_k and s. Computationally this will generally require the construction of a row in an updated LP tableau and a table look-up with interpolation.

It so happens that s can be eliminated between (1) and the Gomory-Johnson inequality, so that one again arrives at a final inequality involving the t_k alone.

There are, then, actually three states at level l. Each yields, from the associated linear programming tableau, one or more inequalities. The branch variable k^*, then, could be chosen with reference to any of these three different states and is in fact chosen from the first state.

In our computational work we have, because of time limitations, refrained from some of these options. For example, we construct Gomory-Johnson inequalities only from the first LP tableau at level l, i.e., from the tableau determining the first state. See the next two sections for a description of the three LP solutions. The associated LP tableaux can be visualized easily.

We choose the branch variable, trivially, as the first fractional variable, characterized by the index f. This

makes the algorithm somewhat similar to a single-branch BB algorithm. But the other options do exist and may very well be worth exploring further.

• *Linear programming solution*

At a particular node of the search, let the state be given by (E, Z, F). For the time being, F is not yet partitioned further, nor does the further partitioning play a determining role as in the general case of state enumeration. We may assume that the problem is feasible and nonintegral. (If the program is not feasible a backup in the search is indicated.) The solution is then uniquely determined by the index $f \in F$ of the one variable that becomes fractional.

As a matter of fact, F is partitioned into the sets L (consisting of all indices in F that are to the left of f), $\{f\}$, and R (all indices of F to the right of f):

$$F = L + \{f\} + R. \tag{5}$$

In the LP solution, the variables t_k, $k \in L$, have value one and those with $k \in R$ have value zero.

Example (Note that the p_k/b_k are ordered):

$$p = (15, 27, 10, 15, 18, 10, 41, 32, 62, 70),$$

$$b = (11, 25, 10, 15, 20, 12, 50, 40, 80, 100), \qquad r = 74$$

$$E = Z = \varnothing, \quad F = K.$$

$$\text{LP solution}: f = 5, \quad L = \{1, 2, 3, 4\}, \qquad R = \{6, 7, 8, 9, 10\},$$

$$t = (1, 1, 1, 1, 13/20, 0, 0, 0, 0, 0).$$

• *Three integer solution approximations*

Not all solutions need exist, in general. The solution of i) may be good, and ii) and iii) may occasionally not give an improvement. But usually the simple heuristics of ii) and iii) will be found effective.

i) A trivial solution is obtained by rounding down t_f.
ii) Round t_f to zero. Then scan right and try to refill the knapsack at unit levels from variables in R.
iii) Round t_f to one. This renders the solution infeasible.
   1. Set $j = f$, $J = L$.
   2. If J is void, abandon the search without solution. Otherwise, scan left (from j) and set, consecutively, all t_k, $k \in J$, one at a time, to zero. If there are feasible solutions, take the best one and stop.
   3. If there are no feasible solutions, set t_k to 0 for the rightmost index in J. Drop this rightmost index from J and set j to the new rightmost index of J (if possible). Go to 2.

With each of the three integer solution approximations one can associate a linear programming solution by giving the first variable from the left in F a fractional value, which ensures that the knapsack is full.

Example: In the case of the example given above, one has three LP solutions, z_λ being the objective function for an LP solution and z that for an integer solution):

$$t^1 = (1, 1, 1, 1, 13/20, 0, 0, 0, 0, 0), \qquad f^1 = 5, \quad z^1{}_\lambda = 78.7,$$

$$t^2 = (1, 1, 1, 1, 0, 1, 1/50, 0, 0, 0), \qquad f^2 = 7, \quad z^2{}_\lambda = 77.82,$$

$$t^3 = (1, 1, 3/10, 1, 1, 0, 0, 0, 0, 0), \qquad f^3 = 3, \quad z^3{}_\lambda = 78.$$

The three integer approximations are the same with $t_5 = 0$ for t^1, $t_7 = 0$ for t^2, and $t_3 = 0$ for t^3. The integer objective functions are $z^1 = 67$, $z^2 = 77$, $z^3 = 75$. The second solution happens to be optimal.

It would probably not be very difficult to devise somewhat more elaborate heuristic rules.

• *Benders inequalities*

Instead of giving a general discussion of Benders inequalities and their use in enumerative programming we refer the reader to Refs. 9 or 10. It should suffice to note that, for any optimal LP tableau, one may look at the top row which expresses z as a linear combination of the nonbasic variables $(k \in N)$. Among these variables are, in general, some integer variables y_k,

$$z = z_\lambda - \sum_{N \cap K} \tilde{h}_k \tilde{y}_k - \cdots. \tag{6}$$

The summation is over the nonbasic variables only. The $\tilde{h}_k$ are essentially the reduced costs. At optimality, all $\tilde{h}_k$ are nonnegative for $k \in F$, but not necessarily for $k \in E$ or $k \in Z$. The $\tilde{y}_k$ stand for $y_k (k \in R + Z)$ and for $\bar{y}_k = 1 - y_k (k \in L + E)$. If z^* is the best-known value for an integer solution, then an improvement can be found only for values of the $\tilde{y}_k$ such that

$$z_\lambda - \sum \tilde{h}_k \tilde{y}_k > z^*. \tag{7}$$

Therefore, rewriting (7) in terms of the original variables y_k, or rather t_k for the knapsack problem, one has a linear inequality over the t_k. Fortunately, the knapsack problem affords the rare opportunity of writing the inequality in the closed form:

$$\sum_{N \cap K} h_k t_k < z_\lambda - z^* + \sum_{(L+E) \cap N} h_k, \tag{8}$$

$$h_k = (-p_k + b_k p_f / b_f).$$

The summation is over all nonbasic variables that are either fixed at one, or are at one in the first LP solution, i.e., are to the left of the index F that determines the first LP solution.

In general, one obtains such an inequality for each of the three linear programming tableaus. For the knapsack problem one always has a value for z^*. In a certain sense, L and R and $\{f\}$ play the role of $F1$, $F2$, and $F3$ of the general state enumeration algorithm.

• *Gomory-Johnson inequalities*

The knapsack problem is also a good vehicle for testing some of the important new ideas of Gomory and Johnson [7], to whom we are grateful for making this material available before its general dissemination. We also wish to acknowledge the generous advice of Ellis Johnson, given freely in many discussions. We shall give only a brief description and suggest Ref. 8 for a rather detailed exposition of the basic ideas.

Starting with a linear programming solution with fractional index f, and corresponding index sets E, Z, L, R, $\{f\}$, one may write the well-known group problem, the asymptotic problem of Gomory, as follows:

Let $K1 = (L + E) \cap N$, $K2 = (R + Z) \cap N$. Then

$$\min -\sum_{K1} h_k t_k + \sum_{K2} h_k t_k - (c_f/b_f)s,$$

$$-\left\{\sum_{K1} b_k \bar{t}_k + \sum_{K2} b_k t_k\right\} \Big/ b_f + s/b_f = v \pmod 1, \qquad (9)$$

$$t_k \in \{0, 1\}, \quad k \in (L + R) \cap N, \qquad s \geq 0,$$

in which the complementary variables $\bar{t}_k = 1 - t_k$ have been introduced for $k \in L + E$, and v is a correspondingly modified right-hand side of the constraint. We have chosen to treat the original problem (1) as a minimization problem, min $c_k t$, where c_k is identified with $-p_k$.

Gomory and Johnson show in Ref. 7 how one may obtain valid and relatively strong inequalities of the form

$$\sum_{K1} \pi_k \bar{t}_k + \sum_{K2} \pi_k t_k + \pi^{\pm} s \geq 1, \qquad (10)$$

by taking the faces of tabulated corner polyhedra for cyclic groups, (see Gomory [11]) and by "interpolating" in various ways (see [8] for details). The numerical results below were obtained for the simplest possible case involving the two faces of a cyclic group of order two. One may well expect that the utilization of higher-order groups, not in principle very much more complicated, could lead to substantially superior results.

After the π_k and $\pi^{\pm}$ have been obtained, one may in the case of the knapsack problem eliminate the slack variable s between (1) and (10) to arrive at an inequality over the t only. We also use the obvious inequality, the ceiling test,

$$\sum_{K} p_k t_k > z^*, \qquad (11)$$

which is of the same form as the Benders inequalities.

Numerical results for eight methods

• *Algorithms*

In order to study the comparative merit of algorithms and the utility of approximations and inequalities, we implemented eight algorithms within a modular system of subroutines:

1) A branch-and-bound algorithm of multibranch type, i.e., having as many pending nodes as are generated, stored, and retrieved according to largest objective function value. The integer solution approximation i) is used. No "penalty" computation is made of the type customary in branch-bound codes.
2) The same as in 1) plus "group penalties", computed from a cyclic group of order seven at the suggestion of Ellis Johnson. Customary penalties could be expected to be weaker. This relates to only one pivot step in the rounding up or rounding down of a basic variable, whereas the group penalties give a good measure of the total cost of rendering that basic variable integral.
3) The state enumeration algorithm with approximation i) and corresponding Benders inequalities.
4) State enumeration with approximation i) and Gomory-Johnson inequalities (computed from the cyclic group of order two) as described earlier.
5) State enumeration with approximation i) and Benders as well as Gomory-Johnson inequalities.
6), 7), 8) Same as 3), 4), 5) except that all approximations i), ii), and iii) are used.

• *Numerical results*

In Table 1 we give, for each method, results for 12 knapsack problems of small size (10 zero-one variables only). A number of 75- and 100-variable problems were also run and did not cause any particular difficulties. The 12 problems consist of three different sets of constraint data, each of which was run with the four right-hand sides 74, 80, 139 and 250.

When there is one entry in a table column location, it denotes the number of linear programs that were executed during the solution process. A second entry situated just below the first gives the number of inequalities generated and utilized. Where two numbers are located below the first entry, they give the number of distinct Benders inequalities and the number of Gomory-Johnson inequalities, respectively.

Rows 1 and 2 in the table are a measure of the basic effectiveness of BB programming, primarily for comparison purposes. Results in rows 3, 4 and 5 are representative of state enumeration. We believe that this comparison is fair. Methods 3 to 5 use inequalities, whereas 1 and 2 do not. However, the use of inequalities is an essential

Table 1 Twelve knapsack problems, resolved by methods 1 to 8 (See text for explanation and interpretation of column entries).

Problem	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*
Objective function	*77*	*80*	*127*	*215*	*79.654*	*84.015*	*147.42*	*260.97*	*73.55*	*73.55*	*145.65*	*219.20*
METHOD												
1. Branch-and-bound	26	40	54	30	16	32	30	96	18	18	18	16
2. Branch-and-bound + group penalties	6	14	18	10	14	32	30	78	16	18	14	16
3. State enumeration, (i)	11	27	34	14	10	11	12	62	15	24	8	10
Benders	3	5	5	5	3	3	2	6	2	2	2	2
4. State enumeration, (i)	13	16	20	16	7	6	12	26	2	2	1	2
Gomory-Johnson	20	22	26	22	7	9	13	40	3	3	3	3
5. State enumeration, (i)	10	14	11	8	5	4	7	21	2	2	1	2
Benders, Gomory-Johnson	5,16	5,19	5,18	5,13	4,7	4,6	3,8	5,35	2,2	2,2	2,2	2,2
6. State enumeration, (i) (ii) (iii)	2	8	5	3	2	10	7	43	6	13	8	18
Benders	4	4	5	3	4	4	4	11	3	3	2	3
7. State enumeration, (i) (ii) (iii)	7	9	9	4	4	4	8	23	1	1	1	1
Gomory-Johnson	8	11	11	7	5	5	7	37	3	3	3	3
8. State enumeration, (i) (ii) (iii)	2	6	4	2	1	3	3	16	1	1	1	1
Benders, Gomory-Johnson	4,3	5,10	5,6	3,4	4,2	6,6	4,6	13,32	3,2	3,2	2,2	3,2

feature of state enumeration, just as the use of a tree with pending nodes stored in a table and the selection of a best pending node for further processing is characteristic of branch-and-bound programming.

The relative expenditure of computational effort is difficult to assess, depending on the weight that one wishes to give to storage requirement versus purely computational requirement. Row 2 represents a rather good BB scheme. The results of Ref. 8 seem to suggest that inequalities, especially of the Benders type, are only modestly useful in BB programming if used as they are in state enumeration. (Used differently, in auxiliary linear programs over a set of inequalities, they can be quite effective). As a consequence we believe that the results of the table are not biased and that the good performance exhibited in row 5 is indicative of a considerable potential for state enumeration.

The rows 6 to 8 represent what can be done when one exploits the specific structure of a problem. In the present case, this exploitation is effected by construction of approximating integer solutions. The effect is drastic, even though no particular ingenuity was expended on this. Better approximations can undoubtedly be found at modest additional computational cost. The lesson is (and this is reinforced by similar experiences with special structures in other areas) that it is essential to take problem structure into account as much as is at all possible. Examples of these experiences may be found in studies of set covering and plant location [9], where additional references may be found.

Problem structure can, of course, be accounted for in many ways, and much expertise can be brought to bear upon it. Within the enumerative framework presented here, very simple devices were employed, and effectively so, as the last entries of the table indicate. In our opinion, the state enumeration approach with inequalities, once established in a program, makes the exploitation of special structures easier than might otherwise be the case.

References

1. M. M. Guignard and K. Spielberg, "Search Techniques with Adaptive Features for Certain Integer and Mixed Integer Programming Problems," *Proc. IFIPs Congress*, 1968.
2. M. M. Guignard and K. Spielberg, "The State Enumeration Method for Mixed Zero-One Programming," IBM Technical Report No. 320-3000, Philadelphia Scientific Center, Feb. 1971.

3. P. C. Gilmore, and R. E. Gomory, "The Theory and Computation of Knapsack Functions," *Operations Research* **14,** 1045 (1966).
4. D. Fayard and G. Plateau, "Une Méthode Enumerative pour les Problemes de Knapsack à Variables Bivalentes," Publication 30, Lab. de Calcul, Univ. de France in Lille, March 1971.
5. K. Spielberg, "Plant Location with Generalized Search Origin," *Management Science*, **16,** 165 (1969).
6. C. E. Lemke, H. M. Salkin, K. Spielberg, "Set Covering by Single-Branch Enumeration with Linear Programming Subproblems," *Operations Research* **19,** 998 (1971).
7. R. E. Gomory and E. L. Johnson, "Some Continuous Functions Related to Corner Polyhedra," IBM Research Report RC3311, Feb. 23, 1971.
8. E. L. Johnson and K. Spielberg, "Inequalities in Branch and Bound Programming," NATO conference, Helsingor, 1971. See IBM Research Report RC-3649, December 1971.
9. M. L. Balinski and K. Spielberg, "Methods for Integer Programming: Algebraic, Combinatorial and Enumerative," *Progress in Operations Research*, Vol. 3, Editor, J. Aronofsky, John Wiley & Sons, Inc., New York 1969.
10. J. F. Benders, "Partitioning Procedures for Solving Mixed-Variables Programming Problems," *Numerische Mathematik* **4,** 238 (1962).
11. R. E. Gomory, "Some Polyhedra Related to Combinatorial Problems," *Linear Algebra and its Applications* **2,** 451 (1969).

Received November 22, 1971

K. Spielberg is located at the IBM Data Processing Division Scientific Center, 3401 Market Street, Philadelphia, Pennsylvania 19104. M. M. Guignard is Maitre Assistante, Computing Laboratory, Department of Computer Science, Electronics, Electrotechnics and Automata, University of Lille. Dr. Guignard also served as consultant to the IBM Data Processing Division Scientific Center.